ANONYMOUS COMMUNICATIONS ON THE INTERNET

Ying Wang and Partha Dasgupta
Dept. of Computer Science and Engineering
Arizona State University, Tempe, AZ
{yingw, partha}@asu.edu

Abstract

For many internet applications, it may be advantageous or essential and even critical to protect the identity of participants. Anonymous communication strategies mostly shield the sender of messages from identification. This paper presents a protocol for anonymous communication over internet that can also provide receiver anonymity. Our system is designed to provide anonymity under a rather strong adversarial model in terms of identification, anonymity and resilience to collusion, along with low latency of data delivery and the link utilization. We use a tree-like overlay network, composed of nodes that are interested in mutual anonymity. We present a description of the protocol, an analysis of its anonymity and communication efficiency, and evaluate its performance using message-level simulations.

Keywords

Communication Security, Authentication, Cryptographic Protocols and Application

1 Introduction

With the rapid growth and public acceptance of the Internet as a means of communication, anonymity has become an essential requirement for many Internet applications [1]. Anonymity protects the identity of one or both endpoints of a communication [2]. *Sender anonymity* protects the identity of the original sender from the receiver(s) of the message and other third parties. IP multicast can be considered as an example of practical sender anonymous communication, since a sender of packets is hard to be recognized. *Receiver anonymity* protects the identity of the receiver from the original sender and third parties. A newspaper is an example of receiver anonymous communication where indented recipient for a particular article (most subscribers are not interested in reading everything) is impossible to identify. Sender-Receiver anonymity protects the identity of both the sender and the receiver from each other and other third parties. Some Internet forums (with fictitious screen names) can be considered a weak example of sender-receiver anonymity where no one knows who is the sender and who are the persons interested in the content of a post.

We present a system provides all three kind of anonymity described above, specifically, our system supports bi-way anonymous communications. Our system is built on top of peer-to-peer networks which is completely distributed, self-organized, and scalable with thousands of active participants may communicate simultaneously. Our system can provide high level of anonymity while reducing the latency of data delivery and the link utilization. We achieve the anonymity and reductions by making use of optimized application layer multicast communication. The number of peers joined to a multicast tree is dynamic and unknown to peers, and these properties that make multicast useful for anonymity. Also, our system is designed to against passive and active attacks from outsider nodes and insider peers.

In our system, the sender anonymity is achieved by making everyone relay messages, such that when Alice sends a message no one knows whether Alice is the originator or the relaying person. This leads to a collusion attack. That is if Alice relays messages from Bob to Carol as well as originates messages and sends to Carol, then if Bob and Carol collude they can identify messages originated by Alice. We want a higher degree of collusion protection.

Recipient anonymity is also important. When Alice sends an anonymous message under normal e-mail model she knows the recipient and would like to keep her identity secret. But using Usenet (or other Internet forums) as an example, sometimes the information generator may not know the receiver, and the receiver may not want the generator to know. In case of Internet forums, the sender is anonymous to the receiver and the receiver is anonymous to the sender but the *forum host knows both* (at least the IP Addresses of both). Our anonymity guaranties are stricter – no entity must know anything about the senders or the receivers, as well as collusion should reveal very little information.

The basic approach is that we first initialize the system by building an efficient multicast tree in a decentralized manner, and then senders send messages by broadcasting it through the multicast tree with itself as the root. Any peer sets up its own filter to get messages it interested in. In that way, both sender and receiver anonymity is achieved. Also the structure of the tree has a high degree of fan out at each node, making sender anonymity "*k-collusion-free*" where k+1 is the number of nodes that are

neighbors of the sender. That is at least *k* nodes have to collude before the sender is compromised.

In Section 2, we overview related work. We describe our protocol in detail in section 3. In section 4, we present various attacks and analysis the anonymity provided by our system. In section 5, we analyze the performance of the protocol. We show and discuss the results from our message-level simulator in Section 6. We offer concluding remarks in Section 7.

2 Related Work

Both DC-net [3] and Xor-Trees[4] provide sender-, receiver-, and sender-receiver anonymity. Users send encrypted broadcasts to the entire group thus achieving receiver anonymity and only one user can send at a time.

Both Onion ring [5] and Crowds [6] only provide sender anonymity. They are both composed of a fixed set of routers, but the path selection methods are different. The In onion ring, the originator makes a path through the rerouting network [7] and in Crowds, the path is chosen randomly on a hop-by-hop basis [1].

Tarzan [8] is a system designed to provide anonymous communication at the IP-level [2]. All participants are potential rerouters and there are no centralized components [2]. They system uses dummy traffic to make traffic analysis harder. The path is chosen at random at the initiating endpoint [2].
Requesters in Tarzan are required to have knowledge of a significant portion of the network [9]. Such architecture imposes heavy workload on the participant nodes. Also Tarzan achieves anonymity through special IP tunnels built randomly by participating nodes, different from that, in our approach, we broadcast messages to all participants, so that the receiver anonymity improves since there is no way to know who actually received and read the messages.

P5 [7] allows secure anonymous connections between a hierarchy of progressively smaller broadcast groups and allows individual users to trade off anonymity for communication efficiency. Because the P5 logical broadcast hierarchy is a binary tree which is constructed using the public keys instead of basing on physical connection and network resource usage, the resulting tree may be very inefficient with respect to end-to-end delay and link usage. Further, to construct of the tree, peers need to consult an "oracle" which maintains an up to date list of channel memberships, which is prone to single point of failure.

3 Our Solution

3.1 Structure

The core idea in our approach is a participant group, structured like a modified tree (with interconnections at the leaf level). The use of a broadcast tree is reminiscent of the old *Usenet* bulletin board, which was a store and forward distribution network for messages. Usenet provided receiver anonymity.

In Usenet, every message received by a node from a neighbor is forwarded to all other neighboring nodes. A generated message is sent to all neighbors. We follow a similar technique. The number of neighbors at each node determines the collusion proof degree of the node.

The set of participants are formed by each node joining a community and is similar to P2P systems. When a message is received by a node, it knows one of the members of the group sent the message but not which member. Hence, the larger the group is, the higher the anonymity could be.

3.2 Topology

In our system, we organize all the participating peers into an application layer multicast tree, and each leaf peer connecting a number of randomly selected leaf peers. That is for intermediate peers, each peer connects to a parent, a number of children, and for leaf peers, each peer connects to one parent, and a number of friends. As shown in Figure 3-1, all the peers form a multicast tree, and each leaf peer connected to at least two other leaf peers.

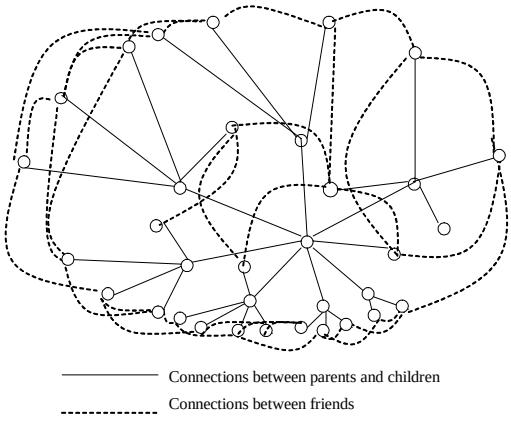

Figure 3-1. System Topology

In order to improve the network efficiency and reduce system communication cost, we adapt [9]'s ideas on building a tree based on the network condition, including latency, and bandwidth. And the goal of tree building is to mimic the underlining physical network connections, so that the link usage and the latency could be minimized. The reason, for which we add friends, is to keep each leaf peer connect to at least *n* friends to provide a desired level of anonymity (Details discussed in section 3.2.2). The more number each peer connected to, the higher level of anonymity.

3.2.1 Initialization

In our solution, we assume that each peer knows a list of other peers from bootstrap, and the peers in the list are

called neighbors of the peer. We achieve the tree building in the following three steps. First, the source sends out tree-creating-message to all of its neighbors. Second, for each peer that receives the tree-creating-message will broadcast the tree-create-message with TTL(Time To Live) equals to two, and send join message to its parent. Third, for each peer receives joining request will follow a tree-joining algorithm. If the peer has the capability to accommodate one more children, then accepts it, otherwise, it will compare its worst child with the requesting one, and accept the better one, reject the worse one. Peers can always join/rejoin the tree at any time from any peer in the tree.

To add friends, peers are required to give a list of peers they know to their children, so that their children could include the list into their list and then randomly pick peer from lists to pass down until the leaf peers. A leaf peer searches for friends by sending out be-friend request. Peers, who receive a be-friend-request, could either accept it by adding the requester to its list, or reject it and send back a list of peers they know, from which the requester could choose to become friend. The requester either adds a friend upon receiving the acceptance or selects and send s request to another peer until the lower bound achieved.

3.2.2 Maintenance

To maintain the system topology, we use two heartbeat timers at each peer in the tree. One timer is used to send out heartbeat message periodically and the other is for detecting connections to its neighbors (parent, children and friends). If it does not receive its parent's heartbeat on time, then it will start selecting new parent. If it does not receive its children or friends' heartbeat, delete the child/friend from its children/friend list.

3.2.3 Optimization

In the process of optimization, the overlay topology mimics the real physical network connections so that the difference between them is minimized. The peers execute the optimization algorithm periodically to adjust their positions in the Tree dynamically to achieve better system performance and scalability. The heartbeat messages contain the children list of the parent, the ancestor list of the parent, etc. The peer follows the grid optimization algorithm to find a better parent upon receiving the heartbeat message from its parent.

The basic idea of the optimization algorithm is that: first, if there is no penalty for the peer to get the message from its sibling, that's the physical path from the parent to the peer goes through the sibling, then we use the sibling as the peer's new parent, then we can reduce the link stress between the parent to the sibling, that means we saved network resources. Second, if it is possible for the peer to move closer to the source in the tree, that is if the capacity of its ancestor is not full, it's more efficient to add the peer under its ancestor than under its parent. Peers receive join requests from siblings or decanters will follow tree-joining algorithm.

3.3 Anonymous Communications

Anonymous communications are achieved through the cooperation of all participant peers. Each peer provide four basic functions, based on these basic functions, the system provides four system functions, and then any peer can use the system for anonymous communications as they desire.

3.3.1 Basic functions

- Send

Any peer can be the originator of a message. For each message to be send, the message should first be encrypted and padded to the system pre-define size. A section random number and a message sequence number are used for the receivers to re-construct the messages. The intermediate peers send their messages by broadcasting it to its parent and all of its children. And the leaf peers send their message to a randomly chosen friend.

- Receive

In our system, all the messages are sent to all of the peers. Each peer sets a filter to get the interested messages, so no one knows which peers get which messages.

- Relay

All peers in the system are responsible for message relay. For any message come in, the peers should broadcast it by sending the message to its parent and all of its children except the one from whom received the message.

- Pass

Pass is the function preformed by leaf peers only. Any leaf peer, who gets messages from its friends, makes a random choice either to pass the message to another friend or send the message to its parent. There is a system wide parameter, the forward probability, which indicates the probability with which a peer will choose to pass.

3.3.2 System functions

- Message Broadcasting

Message broadcasting is used to broadcast a message to all of the peers in the system. Any peer can start message broadcasting, who ever receives the message continuing broadcast it until all leaf nodes are reached, i.e. all the nodes receive the message.

- Secure channel for leaves

The secure channel is used for the leaf peers to send messages without expose their identity. If the leaf peer broadcast message by sending it to its parent, its will be exposed since there is no children for leaf peer. To overcome this, the leaf peer sends its messages though a secure channel to another leaf peer and that leaf peer will

broadcast the message. To build the secure channel, the leaf peer passes the message to other leaf peer, until a leaf peer decide to broadcast the message.

- Cover traffic

To provide anonymity against a global eavesdropper, we use cover traffic to maintain peers' traffic patterns, i.e. the traffic pattern should be statistically independent of it originating data traffic. In our system, peers send messages to their parent and children and friends at a system pre-defined rate. When a peer wants to send a message, it exchanges the random dummy message with the signal message, and there is a fixed bit indicate whether it's a dummy message or not. The messages are encrypted with the public key of the next hop, so it's not possible for the outsiders to know which messages are dummy messages. For all signal messages coming in, peers first decrypt and check the integrity, then re-pad and re-encrypt it, last re-order the messages in the outgoing buffer and send them one by one.

- Tree Adjustment

The multicast tree is build to mimic the underlining physical connections; the tree structure will remain relative stable once the structure is settled down. The attackers could figure out the neighbors of a targeted peer, and compromise all its neighbors, which will result in the expose of message sending from the targeted peer. Besides the instability of the peers which goes on and off constantly, we design a protocol to let the tree adjust itself periodically, so that the tree structure is not predictable at the degree that the attacker can not compromise all neighbors of a peer within that adjustment period.

3.3.3 Anonymous Communications

- Sender-receiver anonymous communication

Sender-receiver anonymous communication is achieved by performing message broadcasting if the sender is an intermediate peer and passing through secure channel and then broadcast if the sender is a leaf peer.

If the sender is a leaf node, it sends the message through a secure channel, since the forming peers of the channel are randomly chosen at each step; any peer in it except the sender could not know who the sender is. Once the message is broadcasted, the fist intermediate node could not know who the send is either, it could be any leave node in the tree, and the anonymous set for other intermediate node is even larger. Hence, the communication is sender anonymous. Since every node receives every message, and they filter out everything they want by themselves, it is receiver anonymous.

- Sender anonymous communication

We assume in this case the sender knows the receiver's public key. The sender encrypts the message using the receiver's public key, and then broadcast the message and the message includes the receiver's public key. The receiver filters all messages on the public key field, and thus receives the correct reply. For further anonymity the public key can be changed periodically. How the recipient responds to the message is discussed in the "bi-directional" communication bullet.

- Receiver anonymous communication

The sender broadcast the message, and only interested receiver will read the message and there is no way for the sender to know who received it. The receiver could reply to the send in the sender anonymous way described above.

- Bi-Directional communication

The bi-directional communications can be enabled in several ways. The easiest method is to embed a "reply-nonce" or random number in the message. The responder can embed the same nonce in the reply and the original sender can filter out the message from all the messages received. This however does not provide privacy, and also allows an adversary to group messages and responses. If this is undesirable, the sender can embed a one-time generated public key as the reply-nonce and the responses can be encrypted with this public key. Filtering such messages of course is more compute intensive. Finally, the sender, if she used sender anonymous communication, can put the reply-nonce, encrypted with the recipient's public key in the message. The receiver responds by encrypting the response with the reply-nonce and sending the message out. The original sender will have to decrypt all messages, but has to use symmetric cryptography only. Putting lcear text one-time public keys on responses work, but then messages can be grouped by the one-time public key.

4 Attacks

Anyone outside the peer-to-peer system, it can monitor all the traffic going through the network. Since all messages are encrypted at each hop, the hacker can not know the content of the messages. Since the traffic through the system maintains a stable traffic pattern, each peer maintains a fix incoming/outgoing rate and all the messages are encrypted, there is no way to figure out who is the sender or the receiver even the hacker can monitor all the traffic. Since all the messages are padded to the same size and the padding is changed at each hop, and the messages are re-ordered before passing on, the hackers can not correlate the incoming and outgoing messages. Alteration at the network layer is prevented by integrity checks at each hop, and altered messages will be dropped. A replay attack will re-send an incoming packet and watch for an outgoing packet, a duplicate that will correlate the incoming and outgoing packet [2]. In our system, messages are sent to all the peers in the network through a stable topology, replay can not expose the receiver, or the sender. Also, each peer passing messages at a fixed rate, the DoS attack won't work.

The internal adversary can monitor all the communication between peers and in addition is also trying to compromise the internal peers of the network. An

adversary agent at such a compromised peer can gather information about messages that traverse the peer [1]. The attack that is most likely to compromise the anonymity is the *collusion attack*. In case the sender is an intermediate node, the only way to know which the sender is that the attacker compromise all of its parent and children, and so can know that its children didn't send it the message, yet its parent received the message, then figure out the intermediate node is the sender. For *k* compromised nodes over *n* total nodes, the probability of these *k* nodes is round some nodes is very low. Also the tree is dynamically changing due to the instability of the peers which go on and off constantly, it makes it harder to figure out who are the neighbors of the targeted peer and it leave limited time to compromise all the neighbors of the target peer. In case the sender is a leaf node, the only way to know which the sender is that the attacker compromises all the friends of that node, when there is an outgoing message and no income messages from the friends, the leaf node is the sender. Again, probability that attackers happened to compromised all the siblings of one leaf node is very low and the peer picks new friends constantly. It's harder to compromise all the friends of the targeted peer.

5 Performance Analysis

Since in our approach, all of the messages are sent to all participating peers, performance could be a big concern. However, in order to defend against various attacks, including traffic analysis, message correlation and collaboration, more and more anonymous communication solutions use cover traffic, which means each participating node sends out noise messages at a fixed rate. Comparing to the systems using cover traffic at the same transition rate, our system does not show any performance degrading, because in both kind systems, each node sends out messages to its N neighbors, and for the same level of security, N in both systems should be close. Comparing to other P2P solutions, our topology is built to mimic the underlining physical network connections, so the link usage and the latency in our system will be better than those systems which use the application layer topology directly. From the security point of view, at the same transition rate, we send messages to all the nodes to enhance the anonymity, which could show better anonymity than the systems that only send messages to dedicated receivers, the rate of signal message over noise message will be higher in our system.

Most current solutions only support sender anonymous communications. There are two solutions support sender-receiver anonymous: DC net requires a bus for all the participating nodes, which is not realistic, and Xor-tree requires nodes knowing all the other nodes. Our system adapts their ideas of broadcasting messages to all the participants to achieve anonymous. However, we use application multicast tree which mimic the underling physical connections for broadcasting, which provides anonymity in a cheap, scalable and efficient way.

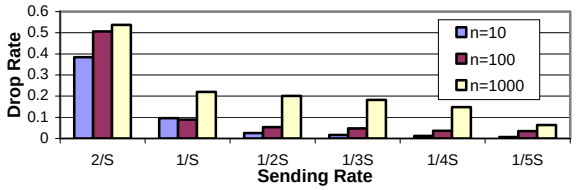

Figure 6-1. Drop Rate vs Sending Rate

Figure 4-2. End-to-end Latency vs Sending Rate

The mix-net systems, like onion routing, crowds, the anonymity is provided through a small, fixed core set of relays [8]. If a corrupt reply received traffic from a non-core node, the relay can identify that node as the ultimate origin of the traffic [8]. Colluding entry and exit relays can use timing analysis to determine both source and destination [8]. In our system, all peers are potential senders, receivers, and relays. Such a scalable design lessens the significance of targeted attacks and inhibits network-edge analysis [8]. Further, mix-net systems are prone to single point of failure or service degrading due to the attacks on a number of relays. Since our system is built on top of P2P system, which is highly distributed, self-organized, the single point of failure will not happen; also it is much harder to attack most of the peers. Further, we take advantage of P2P system's ability to harness a massive amount of resources available on the Internet, the cost of building such system is much cheaper than systems using dedicated servers.

6 Simulation

Through our simulation experiments, we used Georgia Tech [11] random graph generators to generate topologies of the peer-to-peer network. We designed and implemented three basic experiments: 1. Measure system performance as the network size increases. 2. Measure the effect of different number of friends' lower bound. 3. Measure the effect of different buffer queue limit.

In our simulation, we keep the traffic rate fixed by using a timer; at every time out, the peers either send out noise messages or signal messages in the output queue. We assume an unbounded input queue length and a bounded output queue. Output queue limit is enforced, if the output queue is larger than its maximum specified size, messages will be discarded.

First, we measured the system performance as the network size increases. Each bar group in the figure is corresponding to a sending rate. In the "1/2S" case, each peer generates a message at each time stop with probability 1/(2*S), where S is the size of the network. The results (Fig. 6-1) show that the drop rate decreases

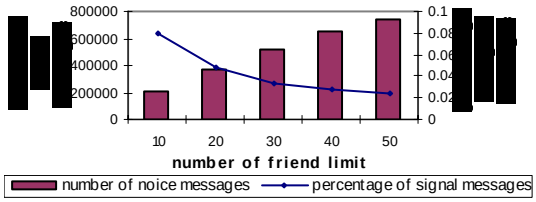

Figure 6-3. Number of noise messages vs percentage of signal messages with different number of friends limit

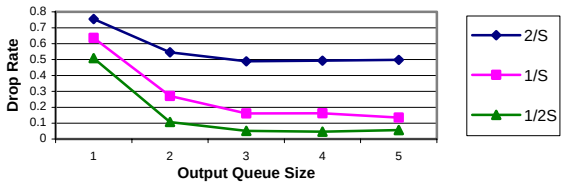

Figure 6-4. Output Queue Size vs Drop Rate

when the sending rate decreases, and the larger the network, the higher the drop rate. The reason is that, with the sending rate increases, the output queues are easily full, and system is closer to saturate, so the drop rate increases.

In Figure 6-2, we show that the end-to-end latency increases as the network size increase and the end-to-end latency decrease as the sending rate decrease. As network size increase, the number of hops between two peers increases, so the end-to-end latency increases. As the sending rate increases, the messages take longer time to wait in the queues, so the end-to-end latency increases. Note, the reason that the end-to-end latency is lower at sending rate is 2/S is that the drop rate is high at this sending rate; messages are dropped in the way to the farther peers. The end-to-end latency is not counted for the messages not reaching the peers on longer paths.

Second, we measure the effect of different number of friends' lower bound. Figure 6-3 shows that with the increasing of the friends' number, the number of noise message increases linearly, and the percentage of signal message decreases. Since only one out of the set number of friends will be chosen to pass signal messages each time, the traffic to other friends is all noise. So more friends peers have, the more noise message sent and then the less percentage of signal messages is.

Third, we study the effect of different output queue limit. In Figure 6-4, we show that the drop rate decreases as the output queue size increase with various sending rate. The reason is that the longer the queue, the larger the buffer the peers can get to store the unsent messages due to heavy traffic going on, and get better chance to send these messages during off-peak period.

7 Conclusion

In this paper, we present a protocol which provides sender-, receiver-, and sender-receiver anonymity. Our system is built on top of peer-to-peer networks which is completely distributed, self-organized, and scalable with thousands of active participants may communicate simultaneously.

From our analysis on anonymity and performance, and various simulations using our message-level simulator, we show that our system provides high level of anonymity while reducing the latency of data delivery and the link utilization. It is not only resilient to various passive and active attacks from outsider nodes and insider peers, but also achieve high level anonymity in a cheap, scalable and efficient way.

Acknowledgement: The authors would like to acknowledge Dr. Dijiang Huang for discussions on anonymous communication and for suggesting the need for collusion prevention.

References

[1]. Yong Guan, Xinwen Fu, Riccardo Bettati, and Wei zhao. An optimal strategy for anonymous communication protocols. Texaz A&M University.

[2]. C. Boesgaard, Anonymous Communication in Practice, Department of Computer Science, University of Copenhagen, Denmark, 2003

[3]. D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Intractability. *Journal of Cryptography*, 65-75, 1989

[4]. Sholmi Dolev and Rafail Ostrovsky. Xor-Trees for efficient Anonymous Multicast Receiption. *Advances in Cryptography* 1997

[5]. M.G. Reed, P.F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. I*EEE Journal on Selected Areas in Communication*, 1998.

[6]. Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 66-92, 1998

[7]. Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5: a protocol for scalable anonymous communication. University of Maryland

[8]. M. J. Freedman, E. Sit, J. Cates, and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. P*roc. ACM Computer and Communications Security*, Nov. 2002.

[9]. A. Mislove, G. Oberoi, A.Post, C. Reis, P. Druschel, AP3: Cooperative, decentralized anonymous communication, *In Proc. SIGOPS-EW, Leuven*, Belgium, Sept. 2004

[10]. J. Jannotti, D.K. Gifford, K. L. Jognson, M. F. Kaashoek and J. w. O'Toole Jr, Overcast: Reliable Multicasting with an Overlay Network, Operating System Design and Implementation, Oct. 2000

[11]. Modeling Topology of Large Internetworks , http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html