ANTI-EAVESDROPPING COMMUNICATION LAYER TO PROTECT AGAINST TRAFFIC ANALYSIS

A Thesis

by

YIPING SHEN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2002

Major Subject: Computer Science

ANTI-EAVESDROPPING COMMUNICATION LAYER TO PROTECT AGAINST TRAFFIC ANALYSIS

A Thesis

by

YIPING SHEN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

[illegible]
Jyh-Charn Liu
(Chair of Committee)

[illegible]
Hoh In
(Member)

[illegible]
Garng Huang
(Member)

Donald F. Friesen
Donald Friesen
(Head of Department)

December 2002

Major Subject: Computer Science

ABSTRACT

Anti-Eavesdropping Communication Layer to
Protect against Traffic Analysis. (December 2002)

Yiping Shen, B.S., Southeast University, China
Chair of Advisory Committee: Dr. Jyh-Charn Liu

In this thesis, we present unicast and multicast protocols to resist eavesdropping and traffic profiling of group communications. At the application layer, we propose a secret-sharing approach for the exchange of shared keys. That is, multicast groups use digital signatures to identify a specific secret-sharing rule, so that nodes in the same group can determine their session keys independently. After the initiation phase to establish group memberships and exchange shared key(s), communicating nodes fragment and shuffle messages into unicast or multicast packets to be transported along different paths. We propose two different transport layer primitives for packet delivery. In the breadth-first approach, packets carrying scattered message pieces are relayed in two stages across group members. For the depth-first approach, group members are configured into multiple rings, each of which is responsible for delivery of one packet to the destination. In both cases, only nodes that have proper keys can decode them. To resist traffic profiling attacks, all nodes keep the inbound and outbound traffic volumes identical via mixed transport of real and decoy packets. Further protection can be added by making the group identifiers secret.

To my Parents and Diana

ACKNOWLEDGMENTS

My advisor Steve Liu served as a reliable sounding board and guiding force in my studies at Texas A&M University. I am indebted to him for his unceasing support during my numerous transitions at Texas A&M University.

I am also thankful to Dr. Hoh In and Dr. Huang for their guidance and good suggestions for my work.

Of course, I am grateful to my parents and Diana for their patience and encouragement. Without them this work would never have come into existence (literally).

Finally, on a more personal note, I wish to thank the following: Di Wu, Yong, Sai, Chunhua, Mingzhang, Tam, Xuyang, Todd (for their excellent discussions and support); Haitao, Danchen, Patrick, Shuihui (for their friendship); Jen and Jim Roselius, Sophia and Andy Chan, Charlie (for their guidance my life); Vishi, Saurabh, Sei-Young, Niyi, Ravikumar, Sanju, Samuel, Vivek, Haixu, Weiming...(for all the good and bad times we had together at RTDS lab).

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

With the broad application of electronic credentials over the Internet for various transaction activities, protection of such high value information becomes critical to the functionality and integrity of the electronic commerce. While cryptographic measures are adequate for most common applications, for highly sensitive applications, such as key management centers, it may also be neccssary to conceal the network traffic patterns. Otherwise, the profile (volumes, peak times, etc.) of the communication traffic may unveil the nature of certain activities, e.g., a command center might be giving an order to its subordinate units. If eavesdroppers can predict presence of high value packets, such as key refreshing messages, they would have a much better chance in cracking the systems.

In "Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems" [1], Raymond presents the traffic analysis problem and expose the most important protocols, attacks and design issues. Our main goal is to protect users against traffic analysis, especially in a critical group communication system. That is, we don't want an adversary that can monitor and/or compromise certain parts of this group communication system and be able to match a message sender with the recipient (sender-recipient matchings).

A related problem is that of network unobservability which attempts to hide all communication pattern. (how many, at what time and to whom/from whom messages are sent and received). Notice that network unobservability implics the incffectiveness of traffic analysis. Whereas message privacy can be obtained using

The journal model is *IEEE Transactions on Automatic Control.*

encryption, it is much harder to protect sender and/or recipient privacy; especially in large open networks. The number of different assumptions and settings is huge which makes it difficult to define and reason about the problem in a rigorous manner. As with many constructions in cryptography, there are efficiency, practicality/security tradeoffs to be made. For example, if efficiency and practicality were not issues, we could broadcast messages in order to protect recipient privacy.

Notice that the problem definition isn't entirely trivial. We can not provide perfect privacy since the number of possible senders and recipients is bounded. So, for example, if there are only two parties on the network, an attacker having access to this information can trivially determine who is communicating with whom. The best we can hope for is to make all possible sender-recipient matchings look equally likely. That is, the attacker's view's (by view, we mean all the information available to the attacker) statistical distribution should be independent from the actual sender-recipient matchings.

Unfortunately, until now there are still no satisfactory definitions or methods providing a solid framework in which to protect against traffic analysis in current group communication systems. At the time of writing this paper, little study has been done on the transport layer in order to protect group communications from traffic analysis [2] [3]. Critical issues on how to secure group communications have been widely studied in the literature. In the supporting layer, there are numerous cryptographic techniques for handling group key management, such as extended Diffie-Hellman key exchange [4], Chiou and Chen 's secure locks based on Chinese Remainder Theorem [5] , secret sharing scheme [6] [7] [8] and key graphs [9] [10]. Moyer [11] proposed evaluation criteria about key management solutions. In application layer, Thomas [12], Butler [13] and Kerberos [14] described typical solutions for distributed authenticity, and Nathalie [15] introduced the secure anonymous group infrastructure.

Reliable multicast (data dissemination) [16] service such as Muse [17], MDP [18], RMTP [19], MFTP [20] have been proposed in order to support the transport layer.

In this thesis, we propose data transport protocols by which it becomes very difficult, if not impossible, for eavesdroppers to determine the interaction patterns in the group. At higher networking layers, a message is fragmented, sliced, and then transported along different paths to reach its destination. By making the overall group traffic patterns uniform and dispersed, our scheme drastically increases the network resources necessary for the eavesdroppers to acquire the message fragments, and crack the complete messages. Depth-first and breadth-first approaches are proposed here to provide different levels of protections, at different performance costs.

CHAPTER II

SYSTEM MODEL

In this chapter, we mainly introduce the system running environments. First, we will give the layered security requirement of the whole communication system. Then we will describe the system components and general operations under two types anti-eavesdropping broadcast algorithms, which we will introduce in next chapter. Based on the system model, we will describe two anti-eavesdropping broadcast algorithms and give cost and security analysis from adversary point of view in next chapter.

A. Layered Security Requirement

In this section, we investigate basic issues on how to create private group communications without unveiling their traffic patterns, based on the four layers of security requirements listed in Figure 1. At the application level, we assume that some existing solutions take care of user authentication and anonymity. The credential of a principal, i.e., a user or machine, is bound to a public key through a public key infrastructure (PKI), such as X.509 or PGP. We assume that the trusted entities set guidelines, certify new principals, and validate the binding process, in addition to other operational details. We note that the authentication authority does not necessarily have the secrets for operations.

Message encryption techniques such as IDEA, DES, RSA, or elliptic curve algorithms provide confidentiality and integrity of the payload at the message level. As for group key distribution and exchange at the supporting level, we assume the use of a secret sharing scheme like that of Shamir [6] and Blakley [7], so that a recipient can recover the message when k-out-of-n of the shares or shadows, become

Application layer (authentication, anonymity, anti-collusion)	supporting layer (group key management)
Session layer (confidentiality, integrity, authenticity)	
Transport layer (anti-traffic analysis)	

Fig. 1. Security requirements of the anti-eavesdropping group communication

available. Different choices for the values of k and n reflect the tradeoff between security and reliability. Although we choose the secret sharing scheme here to implement key distribution, it does not preclude us from using other techniques, e.g., extended Diffie-Hellman key exchange [4], Chiou and Chen 's secure locks based on Chinese Remainder Theorem [5] , and key graphs [9] [10] to manage group keys. For the transport level security designs, we are mainly focused on preventing traffic profiling and eavesdropping. By unifying the communication patterns for the four types of node interactive patterns, namely, point-to-point $(1-1)$, point-to-multipoint broadcast $(1-N)$, multipoint-to-point $(N-1)$ and multipoint-to-multipoint broadcast $(N-N)$, our scheme makes it very difficult to determine the interactions among the group members. By scattering a message into pieces for transport along different paths, we make it very difficult to intercept the complete message.

B. Components and Operations Assumptions

We do not consider any type of node and link failures and assume that the low level networking protocols will maintain the connectivity between group-members at all time. As a result, at the application and session layers, our model assumes a fully connected logical topology in which group members are able to communicate with one another via equally weighted paths. No specific requirements on ordering and

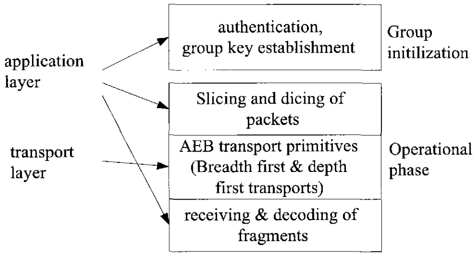


Fig. 2. Main components of the anti-eavesdropping broadcast algorithms

queuing delays of packets are needed for group members and intermediate nodes to deliver packets. For simplicity, we assume that all control and data packets have the same size.

Group communication activities in the proposed anti-eavesdropping broadcast (AEB) scheme are divided into the initialization phase, and the operational phase, see Figure 2. The initialization phase is aimed at establishing the group membership and other related administrative matters. Group members exchange node identifiers and key information. In a group shared key generation procedure, group members also authenticate each other based on a hierarchical or distributed framework. We make use of this authenticated key to identify and communicate with trusted entities to add an additional layer of protection during the operational phase. We note that in this phase the traffic patterns between group members are subject to passive analysis. At successful completion of the initialization phase, all members are authorized and authenticated for group interactions during the operational phase.

CHAPTER III

ANTI-EAVESDROPPING BROADCASTING PROTOCOLS

The operational phase of AEB consists of two main parts. First, data is shuffled, fragmented and transported along different paths using AEB packet transport primitives. A set of communication primitives is designed for dispersing the interaction patterns between nodes, and for making the traffic volumes in all group members symmetric; asymptotically all nodes would have the same traffic patterns. As needed, the payload contents can be encrypted for further protection.

A. Application Layer

Our group communication employs trust relationship among group members as modelled in the Figure 3 below. One or many of these participating nodes are sources and sinks of message exchanges that automatically assume leadership in defining shuffling rules, message fragmentation and group dispersion and interaction patterns. Trusted participating nodes are always active and own a shared group communication key. Trusted forwarding nodes are active in assisting group participating nodes to conceal traffic patterns by exchanging true and decoy messages. Intermediate nodes do not participate in group shared key generation, key restoration, key revocation or final message decryption. Finally, the eavesdropping nodes are not trusted and never possess a shared key.

The three main operations at the application layer include shared key exchanges, packet shuffling and segmentation (at the senders), and reassembly of the segmentation (at the recipients). We describe these three functions in this subsection.

When a node, for example, a broadcast server S, needs to broadcast a secret

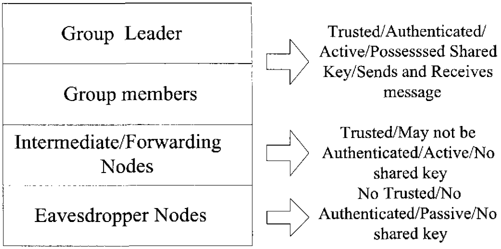

Group Leader
Group members
Intermediate/Forwarding Nodes
Eavesdropper Nodes
Trusted/Authenticated/ Active/Possesssed Shared Key/Sends and Receives message
Trusted/May not be Authenticated/Active/No shared key
No Trusted/No Authenticated/Passive/No shared key

Fig. 3. Levels of node functionality and trustworthiness communication

message M to n nodes, $A = A_1, A_2, \ldots, A_n$, which can be the whole group, or a subgroup within the group. To distribute a key, which is bound to a shared secret among the n nodes, S first sends out the digital signature of a secret-sharing rule, so that only the n selected nodes can determine the session key. Other nodes not belonging to this particular subgroup should only relay packets according to the packet headers. The algorithm is described next.

1. Group Key Generation and Group Management

Group Key Generation Algorithms using secret sharing scheme:

- Input:G, n;
- Output:K;
- $K = F(X, n)$;

1. S randomly selects a bulk of random data X and partitions it into n fragments $G_1, G_2, \ldots, G_n$.

2. S calculates their MD5 digital digests [21] $D_i = H(G_i)$, $i = 1, 2, ..n$.

3. S uses a secret sharing scheme (k, n) [6] with $D_1 \ldots, D_n$ as inputs, and the shared key is K.

4. S sends its time-stamp to synchronize all recipients' clocks and initialize broadcast channels. Recipients update local time-stamps and acknowledge S.

5. Transfers G_i to A_j using an n-complete bipartite matching graph, $i, j = 1, \ldots, n$. G_i is transmitted in packets $G_{i_1}, \ldots, G_{i_n}$. A full handshaking procedure is used to make sure that all recipients get the packets.

6. After S receives all the acknowledgements, it commands all recipients to start broadcast; A_i broadcasts G_i to $A_j (j = 1, 2, \ldots, n, j \neq i)$.

7. After A_i received from all other $n - 1$ nods, it repeats steps 2 and 3 to get the shared key K, $K = F(X, n)$.

After each A_i recovers K, S can send out messages to members in A encrypted by K. Comparing with other group key distribution schemes [4] [5] [9] [10] [22] [23], our algorithm takes advantage of the property of the secret sharing scheme [6] to distribute key-related materials among group members with a balanced traffic pattern. One could increase the protection by repeating the above algorithm a few times to derive the real K.

When an existing member departs from the group, all the shared secrets of this group need be discarded. A simplest resolution is that we restart a new group without including the departing member. The other case is related to addition of a new member. Assuming that the new member is authenticated, we must pass the existing set of shared secrets to the new member. This can be done using a simple client-server protocol as follows. Let W denote the new perspective member, which now has a piece of authentication credential from the PKI. W broadcasts the joint-group request including the PKI-credential. When any of the legitimate group members receives the request, it responds to the new node directly.

To prove to W that it is a legitimate member of the group, each of the group members must own an "evidence function", which can produce an "authentic response" each time when they receive a request. In the simplest form, the authentic response of a responding node can be its own PKI-credential. After W receives one or more authentic responses, it makes a connection request to one responding group member R, and ignores others. After mutual authentication and exchange of a session key K' [24], R can pass all the necessary information to W using K'. From this point on, if the secret sharing scheme needs to be updated due to addition of W, R just needs to use the existing secret sharing scheme to inform others that W has been successfully added to the group. Other group members will need to update all the related computing and communicating processes to accommodate the new node W.

2. Dispersed Broadcast

Conceptually, in our approach we disperse a message along different paths for transport to avoid a single point of eavesdropping. Given that nodes in a group can receive their shared secrets (e.g. session keys), one can use broadcast packets to send unicast and multicast messages but only key owners can decipher the packets correctly. To disperse the traffic patterns among nodes, we use a shuffling and fragmentation technique to randomize the transmission paths of packets of a message. For simplicity,

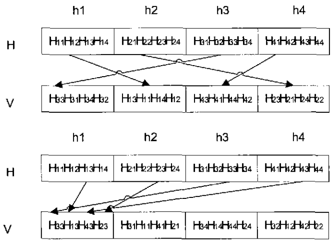


Fig. 4. An example of the double shuffling and slicing of packets in two different ways

we use the perfect shuffling rule to demonstrate the shuffling procedure.

Before we broadcast a message M, we encrypt M using K and shuffle the encrypted message. Suppose $H = EK(M)$ and we use a shuffling rule $R = (r_1, \ldots, r_n n)$, in which position i is mapped to r_i, to shuffle the message fragments $H(h_1, h_2, \ldots, h_n)$, and $h_i(h_{i_1}, h_{i_2}, \ldots, h_{i_n})$. The following figure depicts the shuffling rule (3142) for mapping of H and h_i $\forall i$. Let the shuffling outputs be denoted as $V(v_1, v_2, v_3, \ldots, v_n)$, we send V_i to A_i respectively $(i = 1, 2, \ldots, n)$. Figure 4 shows that one can put pieces of one fragment into one packet, or pieces from different fragments into one packet.

Shuffling rules can be secretly exchanged in a process similar to that of the shared key exchanges. After a node obtains K and R, it will be able to reconstruct the message M according the shuffling rule. The example shown in Figure 5 illustrates our scheme. Here, we have A_1, A_2, A_3, and A_4 in the same group. M is fragmented into G_1, G_2, G_3, and G_4. After S sets up group key K with A_1, A_2, A_3, and A_4, it begins to transfer message fragments to A_1, A_2, A_3, and A_4. For A_1 to recover the message, A_2, A_3, and A_4 need to forward their fragments to A_1. A_1 then uses the R

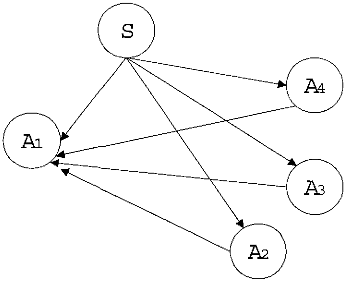


Fig. 5. An example of two-hop relay of message fragments from node S to node A_1

and K to reassemble the fragments in sequence. To break a communication message from S to A_1, an eavesdropper needs to intercept and decode $E_k(G_1)$, $E_k(G_2)$, $E_k(G_3)$, and $E_k(G_4)$, the encryption key K, and the shuffling rule R, failing of any of the steps will not unveil the plaintext message.

B. Transport Layer

To prevent traffic analysis attacks, our approach is to map all unicast/multicast messages into broadcast packets, but only nodes that have the proper session keys can decode the packets into meaningful plaintext messages. We propose two different approaches for the transport layer. The first is a breadth-first approach, and the second is a depth-first. In the breadth-first approach, the source fragments and disperses pieces of the message to group nodes. Intermediate nodes after receiving the "data shares/shadows" relay them to the real destinations. Nodes maintain symmetric traffic by dispersing real and decoy packets to other nodes. For the depth first approach, group nodes are organized into multiple rings, so that data shares are transported along the rings to reach the destinations. Similar to the breadth-first approach, all nodes keep track of the volumes of their inbound and outbound traffic balanced, so that all nodes appear to be symmetric to the eavesdroppers.

1. Breadth-First Protocol

The breadth-first procedure starts after cold-start initiation and group key exchanges are completed. Source S fragments and disperses messages to all participating nodes in the group with predetermined rules for distributing the fragmented messages. In the subsequent phase, all intermediate nodes relay their fragments. An acknowledgement to each fragmented message (including the forwarded message) detects any loss and also serves to balance incoming traffic.

Breadth-First Broadcast Transport Protocol:

- Summary: S fragments and disperses messages M (through broadcast) to n participating nodes $A = A_1, A_2, \ldots, A_n$ (possessing K) and m intermediate relay nodes. All nodes relay message fragments in turn to other nodes to reassemble in the second phase.
- Input: Message M and the shared key K.
- Output: All n participating nodes with K are able to decipher M.

Algorithm Description at source S:

1. Calculate complete message encipher $E_K(M)$.

2. Fragment $E_K(M)$ into P pieces, $F(E_K(M), n, m, P) = G_1, G_2, \ldots, G_P$.

3. First stage relay, choose an arbitrary next hop A_j. Choose a random G_i and encrypt it using K. Transmit encrypted fragment: $S \rightarrow A_j$: $E_K(G_i)$.

4. S retransmits any lost packets, $S \longrightarrow A_j : ACK(G_i)$, if no ACK received before time out.

5. Repeat steps 3 and 4 above for all fragments G_i in $G_1, G_2, \ldots, G_P$.

6. If $P < n$, there are still destinations that did not receive any fragments. Continue the broadcast to reach the remaining destinations.

7. S receives P acknowledgements from all destinations in step 6.

Algorithm Description at receiving nodes A_j:

1. For each $E_K(G_i)$ received, send an acknowledgement to the sender: $A_j \longrightarrow S$: $ACK(G_i)$.

2. Second stage relay, broadcast $E_K(G_i)$ to all A_k, $A_k \in A$ and $k \neq i$, $A_j \rightarrow A_k$: $E_K(G_i)$ $\forall k \neq i$ and $A_k \in A$, and send decoy to others.

3. A_j receives an acknowledgement of the receipt from all other nodes A_k and retransmits if there are any losses during transmissions: $A_j \longleftarrow A_k : ACK(G_i)$.

4. Node participating in shared group communication. If the number of fragments received (and buffered) match the predetermined size (as distributed by original source), an attempt is made to reorder and decrypt the fragments using the shared group key: $D_K(G_1 + G_2 + .. + G_P)$. Ordering and validity of message is verified by successful decryption of the complete message using K.

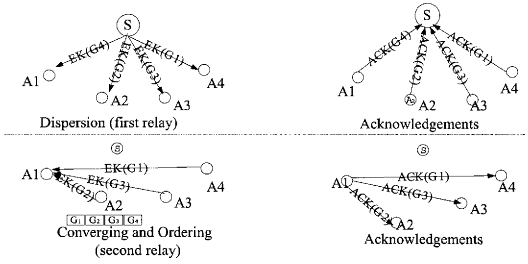


Fig. 6. Operations in a two-stage broadcast process

The protocol operations for the breadth-first approach are illustrated in Figure 6. The figure illustrates two-phase 1-N unidirectional broadcast process with source S and nodes A1, A2, A3 and A4 in a group communication. Shared group key and fragmentation rules are distributed before the procedure starts and is intelligible only to valid destinations.

In a single source $1 - N$ broadcast framework, traffic is made symmetric as the number of inbound and outbound messages is made identical at all nodes. In addition to achieving that, all messages intended for specific destination are mapped into broadcast packets, including the acknowledgements, to conceal the source and destination. Passive eavesdroppers need to know the phase sequence of our process, fragmentation and convergence process and the intended messages before completely making intelligible sense out of the fragments. The same framework functions for a point-to-point $(1 - 1)$ communication between any source S and destination. Only a single destination is able to order and decrypt messages using shared group key K.

Broadcast environment with multiple sources, $N-1$ and $N-N$, are considered as an extension of these basic interaction patterns where a source broadcasts independent of other sources. The degree of protection provided by this scheme increases exponentially with the size of the broadcast group. Suppose that there are k nodes in the group, and the broadcast root S distributes secret information to n members. Even though we assume eavesdroppers know the entire topology, the possibility of the eavesdroppers to know the n group members is: $Prob(G_i|\text{Given } n \text{ known nodes in the broadcast network of } k \text{ nodes})$

$$= \frac{1}{\sum_{i=1}^{k} \begin{pmatrix} k \\ i \end{pmatrix}} = \frac{1}{(2^k - 1)}$$

The total number of packets on the wire across all broadcasts is proportional to the number of fragment pieces P. During first-relay process, the total number of messages on the wire is $2n$ if $P < n$ and $2P$, if $P > n$. The number of inbound and outbound packets is conserved and decoy packets are sent to intermediate nodes if required. During the second-relay process, the total number of packets on the wire is $2n$, as either the node relays decoy packets ($P < n$) or groups more than one message fragment in a single message ($P > n$). The total messages across the wire adding the two stages is then: $\text{MAX}(2n, 2P) + 2n$. Assuming all n nodes take part in the group broadcasting, each node owns a complete copy of the message by the end of the complete process. This results a total message fragment size of: $P * n$.

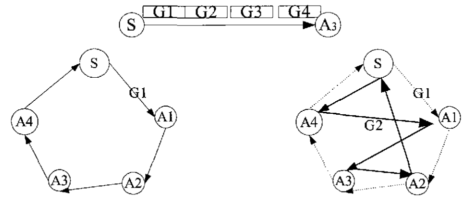
2. Permutation Ring Depth-First Protocol

The basic idea of the permutation ring approach is to organize a group of nodes into multiple logical rings, each of which represents a specific permutation rule for routing and/or shuffling of fragments. For convenience, we adopt the notation of permutation for ring representation, called a permutation ring, which represents a logical interconnection path between group nodes. When S needs to send a message to node A_i, it transmits a shuffled fragment along a randomly configured ring to reach A_i. Unless an eavesdropper knows the shuffling rules R, it cannot decode the message. When S receives a fragmented packet along a particular ring, it knows the fragment has been routed through a ring, and node A has received it.

The transport protocol is termed depth-first, owing to a full packet circulation along the ring regardless of the physical location of the destination in the ring. The basic operations of the permutation ring protocol are depicted as in Figure 7. In this example, S intends to send a message to A_3. The message is broken into four fragments G_1, G_2, G_3, and G_4. The four encrypted fragments can be sent along four different rings to reach A_3.

Based on the ownership of K, a node on the ring can be either a session member, or merely a relay node of the session. It is easy to show that all the $1-1, 1-N, N-1$ and $N-N$ interactions can be implemented using the permutation rings. The number of rings increases at a factorial order resulting in a large search space for the eavesdroppers to crack the packets, not to mention our ability in smoothing the traffic patterns between group members.

A simple routing table could be used in each node to determine the next hop for each ring, given that the table size is reasonably small. Otherwise, ring identifier together with the current node identifier can uniquely be used to find the next hop.



Circulation of fragment G1 across ring

Circulation of fragment G2 across ring

Fig. 7. Illustration of depth-first permutation ring protocol

Topology changes may be propagated across all nodes in two subsequent phases. In the first phase, only the participating group communication nodes exchange authenticity and identification information and update the ring digest. Restoration phase establishes the identity of all nodes, and readjusts permutation sequences and ring digests for all participating nodes. A smallest available node identifier (not used currently) may be reused to save the search space. Deletion of a node results in the removal of the identifier to reflect in the calculation of ring digest. All shared secrets become void on deletion and other group members need to reproduce the shared secrets.

In the second phase, the leader of the current ring (naturally the source S), updates the intermediate (forwarding) nodes of the change in ring digest algorithm and addition of new identifier tags to the topology. Intermediate (forwarding) nodes are trusted but do not take part in restoration phase unlike participating group nodes. Source S acts as the leader and updates the forwarding nodes of any deletions (missing of an identifier) or additions (with new identifier tags) in the ring header. Suitable ring digest algorithm may be calculated and circulated to all nodes by the leader after the restoration phase. This new ring digest algorithm omits intermediate deleted nodes but adds new tags corresponding to newer additions. Next, we discuss the operational protocol of the permutation ring.

Depth-First Permutation Ring Transport Protocol:

- Summary: Each source S fragments and disperses the messages (through broadcast) along a randomly chosen permutation ring that circulates to n participating nodes $A = A_1, A_2, \ldots, A_n$ and m intermediate (forwarding) nodes. All nodes relay message fragments to its immediate next hop in the logical ring. Number of fragmented messages are predetermined and distributed to all participating nodes before the distribution algorithm begins.

- Input: Message M that source wants to send, Shared group key K.

- Output: All n participating nodes with K able to decipher original message M.

Algorithm Description at source S:

1. Calculate complete message encipher $E_K(M)$.

2. Fragment the enciphered message into P pieces, $F(E_K(M), n, m, P) = G_1, .., G_P$.

3. Choose a random permutation pattern $\prod_j = \Pi_{j1}, \Pi_{j2}, \ldots, \Pi_{jn}$ for G_j, $\forall j$. Calculate a ring identifier digest that uniquely identifies the hop sequence $R_j = H_R(\prod_j), \forall j$.

4. $S \rightarrow A_{\Pi_{j1}} : Rj + E_K(G_j), j = 1, 2, \ldots n$. Here, S begins to unicast G_j along $\prod_j$ from the first hop $A_{\Pi_{j1}} \forall j$.

5. $S \leftarrow \prod_j : ACK(G_j)$. S awaits an acknowledgement of G_j from $\prod_j, \forall j$. (S receives G_j from the last hop of $\prod_j$, acknowledging the successful circulation for G_j transmitted across $\prod_j$).

6. Repeat steps 3, 4 and 5 for all the message fragments G_i.

Algorithm Description at receiving nodes A_i (with permutation ring identifier Π_j):

1. Extract the ring digest R_j from the received packet and calculate

   $\prod_j = \Pi_{j1}, \Pi_{j2}, \ldots, \Pi_{jn}$

2. For each received G_j, send a handshake acknowledgement to the previous hop for full handshaking transmission.

3. Unicast the packet $R_j + E_K(G_i)$ to the next hop entry $A_{\Pi_{j,i+1}} : A_{\Pi_{j,i}} \rightarrow A_{\Pi_{j,i+1}} : Rj + E_K(G_j)$. Repeat the transmission if no handshake acknowledgment is received after timeout.

4. Node participating in shared group communication. If the number of fragments received match the predetermined size, an attempt is made to reorder and decrypt the fragments using the shared group key: $D_K(G_1 + G_2 + .. + G_n)$. Ordering and validity of message is verified by successful decryption of the complete message using the shared group key K.

For a given source S and a set of participating nodes that the source uses for constructing a ring, the probability that Q of n (total) nodes become part of the ring is: Prob.(Q nodes out of n take part in ring grouping — Given source S) $= 1/\begin{pmatrix} n \\ Q \end{pmatrix}$

A coordinating set of eavesdropper nodes need to know the ring spaces (rings for all shares) transmitted and shared sequences after knowing K to be able to crack the messages. The degree of protection and hence the probability of continuous (successful) eavesdropping of P message fragments, given that the eavesdropper is not aware of all the participating nodes in a permutation ring communication for a specific source S is $1/[(n-1)!]^P$. Here we assume that S is a member of A, the communicating group. Without having knowledge of the nodes and the rings involved, the eavesdropper search complexity grows at a factorial order. If message-ordering complexity is taken into consideration after a continuous successful eavesdropping of all message fragments, then the overall probability is reduced even further. In this case, the probability of message cracking becomes $1/((P!) * [(n-1)!]^P)$. The total number of messages transmitted on the wire across the ring for a complete message M is proportional to the number of fragments P. If there are n participating nodes, the total number of messages forwarded across the ring becomes $P * 2n$. Each node receives a copy of the entire message fragment set but there are $P * n$ message copies only across the ring, as required.

CHAPTER IV

IMPLEMENTATION AND PERFORMANCE ANALYSIS

We designed and implemented an Anti-eavesdropping Multicast Transporting Protocol (AMTP) on BRICKServer Platform to prevent from network traffic analysis and sniffing for large distributed applications. This protocol is based on Depth-First *Ring* protocol we introduced in the previous chapter. In this chapter, we first introduce our BrickServer platform and protocol run environment, then we give a description of AMTP implementation detail, and finally, we use the networking monitoring software to sniffer each node in order to verify the practical anti-eavesdropping efficiency of AMTP protocol.

A. Why Use BrickServer as the AMTP Run Platform

BRICKServer [25] is based on a robust security model called Process-Based Security (PBS), which is implemented into the kernel of the linux operating system. PBS prevents unauthorized users (external or internal) or programs from creating, modifying, or deleting system resources or data.

The main different between tradition user-based OS and process based OS is that there is a central access control list to control the execution behavior of each program, and all system call that program use in process based OS. The Access Control List is the heart of PBS, which defines the permissions for accessing files and making system calls for each process.

ACL example:

1:EXECUTE /bin/anti-eavesdropping.exe SOCKETS

2:ALLOWED-GROUPS everyone

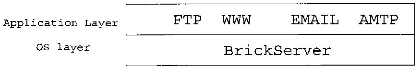


Fig. 8. Secure-guaranteed AMTP service in BrickServer architecture

3:PATH /home/ DELETE

4:PATH /lib/ READ

5:END

Code Explain:

Line 1:this is the full path to the executable name, "SOCKETS" property (defined in following) is the only system call anti-eavesdropping.exe could use.

Some Program rights under PBS:

REBOOT: reboot or power off system

SOCKETS: socket calls, excluding ioctl calls on a socket

SETIO: I/O port control call

CREATEFIFO: create FIFO special file with mknod call

USRSIGNAL: send signal to all processes for a given user call

SYSCTL: the sysctl call

Line 2:you can restrict the ability to execute something by groups or users

Line 3:this allows the program to delete from a user's home directory

Line 4:this gives the program power to link to its dynamic libraries

Line 5:end of definition

AMTP is an application-level multi-cast communication channel used to prevent from network traffic analysis and sniffing for large distributed applications. In order to prevent from other existing or potential security holes (such as buffer overflow or root compromises etc)when running AMTP, we load AMTP module as another

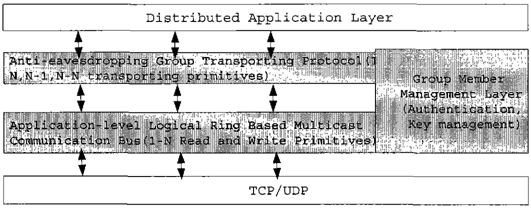

Distributed Application Layer
Anti-eavesdropping Group Transporting Protocol(1-N,N-1,N-N transporting primitives)
Group Member Management Layer (Authentication, Key management)
Application-level Logical Ring Based Multicast Communication Bus(1-N Read and Write Primitives)
TCP/UDP

Fig. 9. Anti-eavesdropping multicast transporting protocol architecture

service similar like www and FTP in existing BrickServer as in Figure 8. Since any program running in BrickServer could effectively resist such security attacks, AMTP program running in BrickServer also could resist potential security holes.

B. Anti-eavesdropping Multicast Transporting Protocol (AMTP) Architecture

1. Layered Design

We use layered design for the whole protocol, each lower level layer provides the communication primitive for its upper level layer, and each layer is an independent module. The lowest layer is networking socket module provided by the operating system, then from down to up, the second layer is ring-based multicast communication primitive(read/write), the third layer is for multicast file/data transport layer based on the second one, and the fourth layer is the distributed application layer that makes all the application traffic resist traffic analysis through calling its low level transport primitive(Figure 9).

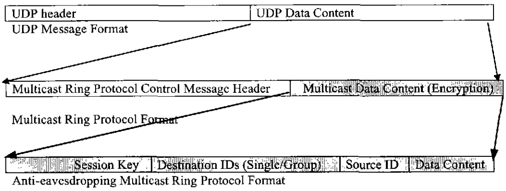

UDP header
UDP Data Content
UDP Message Format
Multicast Ring Protocol Control Message Header
Multicast Data Content (Encryption)
Multicast Ring Protocol Format
Session Key
Destination IDs (Single/Group)
Source ID
Data Content
Anti-eavesdropping Multicast Ring Protocol Format

Fig. 10. Anti-eavesdropping multicast ring protocol payload format

2. Uniform Traffic and Digital Signature to Guarantee Anonymity

Other than resisting the traffic analysis(communication anonymity), our AMTP is also a guarantee sender/ receiver anonymity through adding encryption layer on the transport layer. The behavior of each node in the ring is only forwarding the received packet to the next node(UDP forwarding), and the actual traffic patten in each node is uniform and symmetric(See Figure 10). When a node wants to send some data for some specific receivers in the ring, he only needs encrypt the data using the public key of each receivers. When the data is delivered to each node using our anti-eavesdropping multicast transport protocol, each node in the ring will receive the encrypted data, however, only the receivers designated by the sender could decrypt the encryption data using their private keys respectively. In this process, each receiver couldn't figure out who really sent this data. Thus, sender anonymity is guaranteed. As for receiver anonymity, each multicast communication could satisfy such requirement since the sender couldn't figure out who would be really interested in the multicast message even through each node in the ring did receive his message.

3. Total Ordering and Atomic Multicast Communication to Support E-Transaction

All current group transaction systems have an important requirement for the low level multicast communication: atomic, total ordering. AMTP supports a publish/subscribe paradigm, and implements atomic, totally ordered, group communication. The "atomic" means the property of all or nothing. If a process that multicasts a message crashes before it has delivered it, then it is possible that the message will not be delivered to any process in the group; but if it is delivered to some correct process, then all other correct processes will deliver it. Total ordering means that if a correct process delivers message m before it delivers n, then any other correct process that delivers n will deliver m before n.

Our total ordering algorithm relies on an arrangement of members in a logical ring [26]. When a member sends a message, that message is sent around the ring. Each member receives it, and forwards it to the next member in the ring. As the message travels around the ring, it carries the largest timestamp (a sequence number) of the members it traversed. When the message returns to the sender, the latter knows that everybody has received the message. It then sends a commit message with the largest timestamp that the message encountered on its first trip around the ring. As the members receive commit messages, they deliver the message to the application in increasing timestamp order. Because every message is committed according to a globally unique timestamp (every commit message carries the largest timestamp from all the members), the totally-order can be achieved.

To achieve an atomic communication, we considered the following situations:

(1) Some node misses the data message. The sender of the message can detect such event. After a time period if it does not receive the returned message it sends before, it will send the message again. If it will retransmission this for R times every

T time period until it receives the returned message. In this thesis we assumed that each node cannot fail and communication link never fails, so the message will be received by the node at last after some retransmission.

(2) Some node misses the commit message. Then it cannot be returned to the sender. The same as above, after some time period, the sender will retransmit the commit message until it receives the returned commit message. In this sense, the returned message to the sender can be viewed as a Acknowledgement Message.

These will ensure that all nodes in the group will deliver the message or not.

4. *Member Management in the Ring*

We use a double link list to maintain the logic ring. Each node in the ring only need remember the information of its neighbor nodes. We also set up a registry server to record the information in each ring. When a ring is created with one ring ID, the process created the ring will use UDP to connect the registry server to register such Ring ID. If the ring ID exists already, this node is then "inserted" into the existed "logical ring". And the registry will only keep both neighbor nodes of the header in the ring. The header is commonly the first node that join (create) this ring. Then new member can join into the ring, it will be simply "inserted" into the existed "logical ring" by modifying the links information the others node keep. The existed member can leave this ring anytime. If it is not the "header" of the "ring" whose information is kept in the registry, only modifying the links information(pointer to neighbor nodes) can maintain the "logical ring"; otherwise, we will also need modify the header's information for the ring in the registry.

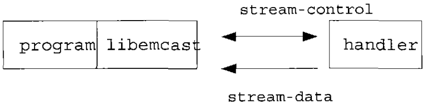


Fig. 11. Emcast interfact with ring handler

5. Uniform Multicast Communication Interface

We imbedded our totally ordering ring protocol into the emcast toolkit [27]. Emcast is a multicast toolkit for distributed/peer-to-peer applications that require multicast communication. It includes the program "emcast", a generic multicast utility (like netcat), and the library "libemcast", a generic multicast library. Emcast supports IPv4 multicast (IM) and can easily support almost any end-host multicast (EM) protocol. The emcast protocols supported are STAR (centralized TCP), Banana Tree Protocol (BTP), and Internet Chat Relay (IRC). Now it also supports our Ring protocol.

A program is compiled with libemcast. Libemcast communicates with an emcast handler using the emcast protocol over two streams, the control stream and the data stream. The control stream is two-way and the data stream is one-way from handler to libemcast. See the diagram below. For example, the handler might be a child process, the control stream two pipes, and the data stream a FIFO.

Libemcast sends requests and receives responses from the handler using the control stream. The handler sends requests using the data stream. The handler can not receive responses using the data stream. If only one stream were used and both libemcast and the handler sent a request, each would think the other's request was a response to its own request, and a malfunction may occur. Using two streams seems the best solution. Here, we wrote our own ring handler in order to interact with emcast.

Usage Example:

1.emcast 234.43.13.42:8765

(emcast joins a IPv4 multicast group)

2.emcast "btp://junglemonkey.net/Monkey Central"

(emcast joins "Monkey Central" on junglemonkey.net using BTP)

3.emcast "ring://dasher.cs.tamu.edu:5000"

(emcast joins our multicast ring channel with id 5000)

C. Experiment and Cost Analysis

1. Testbed Introduction

In a local area network,we use four BrickServers to simulate the real environment of distributed application hosts located in the whole Internet. The difference between these two environments to our test result is the packet transmission delay of each forwarding operation in the protocol.And our objective is to make traffic of each individual node symmetric.And we also make the communication protocol atomic, which means the traffic of each nodes could be still keep symmetric even packet lost existed in real networking. We assume the attacker could sniff the inbound and outbound of TCP/UDP traffic of each individual node in realtime.

2. Traffic Masking and Cost Analysis

In the whole logical ring, only one node(Node 1) sends messages and the other nodes receive messages. In the following four charts Figure 12, we could find the traffic of each node always remains constant in any time interval. It is very difficult for the attacker to find the matching of the sender and receiver in such multicast communication based ring protocol. The real traffic pattern reaches uniform and symmetric in each node through AMTP. However, in the Figure 13, through monitoring the traffic in each node, it is easy for eavesdropper to find the real sender and receiver in the common broadcast communication.

Cost Analysis

Running Cost Analysis: In asynchronous message passing system, we assume that the maximum message delay in any execution is one unit of time and then calculate the running time until termination. Then the time complexity of an asynchronous algorithm is the maximum time until termination among all timed admissible execution in which every message delay is at most one.

For the general broadcast system, the time complexity to deliver a unit message is $O(1)$, and for our AMTP protocol, the time complexity to deliver a message is $O(n*m)$, since each sliced message from original message will circle around the ring. For Breadth-First Protocol, the time complexity to deliver a message is $O(m*h)$. Here, n is the number of nodes in the group, and m the number of sliced message from original message, h is the number of middle hops each sliced message will go through before it finally reaches the designated node.

Buffering Cost Analysis: We assume that the maximum buffering cost in any execution is the total memory buffers used to store and operate one unit of message in all distributed processors. For the general broadcast system, the buffer cost com-

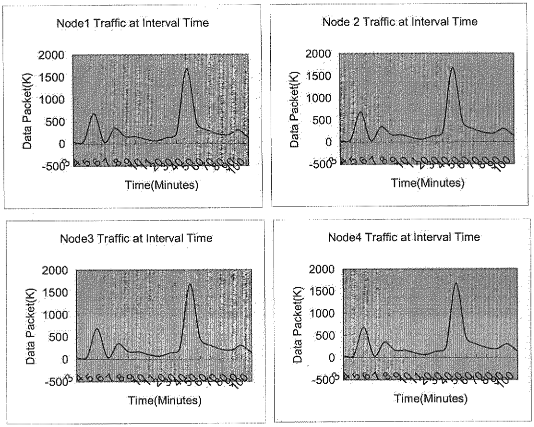


Fig. 12. Interval traffic analysis in AMTP model

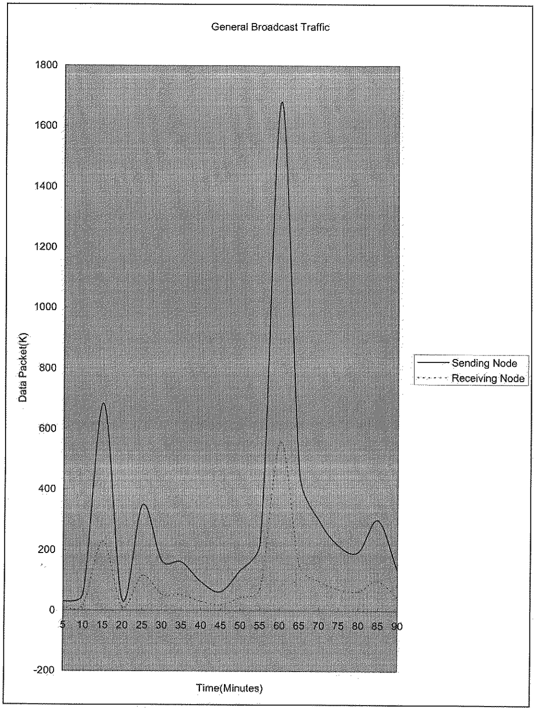


Fig. 13. Interval traffic analysis in general broadcast

plexity to deliver a unit message is $O(1)$, and for our AMTP protocol,the buffering cost complexity to deliver a message is $O(n)$, since each sliced message from original message will circle around the ring, and each node in the ring will store each sliced message. For Breadth-First Protocol, the buffering cost complexity to deliver a unit message is $O(n * h)$. Since each sliced message will be stored temporarily in the middle hops. Here, n is the number of nodes in the group, and m the number of sliced message from original message, h is the number of middle hops that each sliced message will go through before it finally reaches the designated node. We could found the buffering cost is independent of m, the total number of sliced message from an original unit message.

Ordering Cost Analysis: We assume that the maximum ordering cost in any execution is the total operations to recover the original message in the receiver side. For the general broadcast system, the ordering cost complexity to recover a unit message is $O(1)$, since the message ordering is implemented by low level transport protocol, and for our AMTP protocol, the ordering cost to deliver a message is $O(m)$, this result is also applied to the Breadth-First Protocol; since in either case, the total m sliced message will be collected in the receiver side in order to get the original message.

CHAPTER V

CONCLUSION AND FUTURE WORK

In this thesis, we proposed traffic-concealing, anti-eavesdropping communication protocols for secure group communications. By using shared secrets and digital signatures, our scheme need not exchange keys over the network explicitly for data encryption. By using simple shuffling and ordering of message fragments, we disperse the interaction communication patterns among the multicast participants to counter eavesdropping and traffic analysis attacks. The two different types of broadcast-based data transport primitives have been proposed to meet different performance and security requirements. In contrasting the two approaches, the two-hop relay communication may result in a less uniform traffic pattern than the permutation ring approach, but it takes less time to deliver a message. On the other hand, all nodes on the permutation ring would receive the fragmented packets, making it easier for eavesdroppers to acquire complete, yet encrypted messages. Of course, to crack the message, one still must have full knowledge about the permutation and shuffling rules. It takes much more for an eavesdropper to acquire the full messages in the two-hop relay process, as it requires all nodes to be compromised before making intelligible sense of the message. It is of great interest to further expand different types of transport primitives to conceal traffic patterns of group communications.

AMTP provides a very good traffic-concealing, anonymous multicast communication channel for the upper level distributed applications, such as distributed storage system, e-transaction system, or e-vote system. Since AMTP is application-level multicast protocol, it will not rely on any current low level Internet multicast infrastructure.

Although our AMTP communication channel could make the total application traffic uniform in each node in different time phases, it is still possible to detect the non-uniform signal pattern in the physical layer. Even for any message sender or receiver, the packet sending rate and receiving rate were not controlled to reach some concealing pattern. Furthermore, in dynamic network environment, due to the traffic congestion in special situation, there should be some non-uniform characteristic between the nodes, which are near the sending node, and the nodes, which are far from the sending node. Additionally, it is possible for some node in multiple ring at the same time. Is it possible to use control theory to manage such traffic patten in the whole picture? Also, the factor of performance and security are always a tradeoff in design such scheme. All of these factors should be considered carefully in our future work.

REFERENCES

[1] J.F. Raymond,"Traffic analysis: protocols, attacks, design issues and open problems," in *Anonymity 2000 Conference*, Atlanta, GA, pp. 142-145, 2000.

[2] Y. Guan, C. Li, D. Xuan, R. Bettati, and W. Zhao, "Preventing traffic analysis for real-time communication networks," in *Proceedings of MILCOM '99*, Seattle, WA, pp. 24-29, Nov. 1999.

[3] R.E.Newman-Wolfe and B.R.Venkatraman, "High level prevention of traffic analysis," in *Proceedings of Seventh Annual Computer Security Applications Conference*, San Jose, CA, pp. 342-345, 1991.

[4] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," *in EUROCRYPT '94*, vol 3, pp. 213-218.

[5] G.H. Chiou and W.T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, vol.15, pp. 123-129, Aug. 1989.

[6] A.Shamir, "How to share a secret," *Communications of the ACM*, pp. 612-613, 1979.

[7] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of the National Computer Conference*, Austin, TX, pp. 313-317, 1979.

[8] S. Berkovits, "How to broadcast a secret," in D. W. Davies, editor, *Advances in Cryptology - EUROCRYPT 91*, vol. 547, pp. 535-541, Apr. 1991.

[9] Chung K. Wong, M. Gouda, and S.S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 165-169, Feb. 2000.

[10] O. Rodeh, P.B. Kenneth, and D.Danny, "Optimized group rekey for group communication systems," *Network and Distributed Systems Security*, vol 2, pp. 214-219, 1998.

[11] M.J.Moyer, J.R.Rao, and P.Rohatgi, "A survey of security issues in multicast communications," in *IEEE Network*, vol.13, pp. 56-67, Nov. 1999.

[12] Y.C. Thomas and S.S. Lam, "Designing a distributed authorization service," in *IEEE INFOCOM '98,* Lafayette, LA, pp. 272-276, 1998.

[13] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems theory and practice," *IEEE Computer Society*, vol 12, pp. 123-126, 1996.

[14] B. C. Neuman, "Kerberos: An authentication service for computer networks," *IEEE Communications*, vol.32, pp. 33-38, Sept. 1994.

[15] N. Weiler,"Secure anonymous group infrastructure for common and future Internet applications," in *17th Annual Computer Security Applications Conference*, Lincoln, NE, pp. 10-14, Dec. 2001.

[16] K. Obraczka, "Multicast transport protocols: A survey and taxonomy," *IEEE Communications Magazine*, vol 265, pp. 37-44, Jan. 1998.

[17] K. Lidl, J. Osborne,and J. Malcolm, "Drinking from the firehose: multicast USENET NEWS," in P*roceedings of the 1994 Winter USENIX Conference*, vol. 3, pp. 10-14, 1994.

[18] J.Macker and W. Dang, "The multicast dissemination protocol (mdp) framework. Internet Draft," *Internet Engineering Task Force*, 1996, avaliable at http://manimac.itd.nrl.navy.mil/MDP/.

[19] J.C. Lin and S. Paul, "Rmtp: A reliable multicast transport protocol," in *Proceedings of the IEEE INFOCOM'*, Seattle, WA pp. 1414-1424, Mar. 1996.

[20] K. Miller, K. Robertson, A. Tweedly, and M. White, "Starburst multicast file transfer protocol(mftp) specifications," Internet Draft, *Internet Engineering Task Force*, Jan. 1997, avaliable at http://www.globecom.net/ietf/draft/draft-miller-mftp-spec-03.html.

[21] R.Rivest, "The MD5 message digest algorithm," *RFC 1321*, April 1992.

[22] J.K. Jan and C.D. Yu,"Yet another approach for secure broadcasting based upon single key concept," in *Proceedings 25th Annual 1991 IEEE International Carnahan Conference*, Los Angeles, CA, vol 3, pp. 276-279, May 1991.

[23] A. Ballardie, "Scalable multicast key distribution," *RFC 1949*, July 1996.

[24] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol 13, pp.644-654, 1976.

[25] R.A. Ballardie and S.W. Macker,"Online BrickServer technical reports", Available at http://www.thirdpig.com/security.htm

[26] K.P. Birman and T.A. Joseph, "Reliable communication in the presence of failures", *ACM Transactions on Computer System*, vol 5, pp.47-76, Feb. 1987.

[27] D.A. Helder, "Online networking emcast programming library tools," May 2002, Avaliable at http://www.junglemonkey.net/emcast/

VITA

Yiping Shen received his B.S. in Computer Science at Southeast University, Nanjing in 1998. His current areas of interests are applied cryptography and distributed systems. He can be reached at the following address: Texas A&M University, 503 Harvey R. Bright Building, Department of Computer Science, College Station, TX, 77843.

The typist for this thesis was Yiping Shen.