

Secure Group Communication

B.Tech Project, Report

by

Ajit Burad
03005009

under the guidance of

Prof. G. Siva Kumar



Department of Computer Science and Engineering
Indian Institute of Technology

April 15, 2007

Acknowledgments

I would like to thank **Prof. G. Siva Kumar** for his invaluable guidance and encouragement and support for making this project an informative experience for me. I would also like to thank Madhumita A Chatterjee for her guidance.

Ajit Burad

Abstract

Peer to Peer (P2P) communication is emerging as technology having a high potential to provide scalable and flexible structure in communicating, sharing and collaborative applications etc. But to make it secure and more flexible we need a framework which provide key management along with admission control for peer into a group. Key management is important because we need a group key to encrypt all messages with it thus groups secrecy and confidentiality is maintained. Also many group communications require a secure infrastructure that provides multiple levels of access privilege for group members. We describe how a policy language based framework, which supports group communication using role and attribute based access control can be designed. We implemented a prototype chat application based on peer to peer framework, JXTA to demonstrate how a secure group layer which integrates authentication, admission control, authorization, fine-grained access control and key management can be created to achieve secure group communication.

Contents

1	Introduction	3
1.1	What is peer to peer ?	3
1.2	Scenario	3
1.3	Why peer to peer ?	4
1.4	Related Work	4
1.5	Overview	5
2	Admission and access Control	6
2.1	Admission Control	6
2.2	Admission Process	7
2.2.1	Advertising the created groups	7
2.2.2	Request for joining group	8
2.2.3	Authentication of user who wishes to join	8
2.2.4	Voting	8
2.2.5	Issuance of Admission certificate	9
2.2.6	Group Rekeying	9
2.3	Access Control	10
2.4	Member Level Updation	11
3	Key Management	13
3.1	Categorization of Key management protocols	13
3.2	Key management Protocols	14
4	Implementation of Secure MultiChat	16
4.1	JXTA: Peer to Peer framework	16
4.2	Implementation	17
4.3	Functional Components of Framework	19
4.4	JXTA Messages	19
4.4.1	Unicast Messages	20
4.4.2	Broadcast Messages	20
4.5	Policy Language	20
5	Conclusion	22
5.1	Analysis	23
5.2	Future Scope	23
6	Appendix	25
6.1	Screenshots of our Chat Application	25

List of Figures

2.1	New Member Join	7
2.2	Admission process	9
2.3	Role Based Access Control	10
2.4	Member Level Updation	11
3.1	Skinny tree protocol	14
4.1	Integrated Framework	18
4.2	A sample rule file	21
5.1	Framework for Secure Communication control	22
6.1	Found a group advertisement	25
6.2	Chat client	26
6.3	Rating request	26
6.4	Rating Updated	27
6.5	Certificate	27

Chapter 1

Introduction

The secure group communication abstraction provides both point-to-point and secure multipoint communication. This incremental and dynamic growth pattern is not well supported by a rigid server-client based structure. Thus, the natural alternative to a server-client based model is to provide a reliable and secure group communication infrastructure to support a dynamic and scalable peer-to-peer model. Applications like file sharing, online gaming, audio/video conferencing, virtual meeting and discussion forums are example of systems which are organized as peer group. The group is governed by a set of rules that describe the conditions required to be part of a group. Security in such dynamic collaborative groups is governed by membership control, authentication, access control and key management.

This project deals with designing a framework which facilitates peer to peer communication along with admission and access control with security measures. Peer group framework provide flexible structure to communicate, sharing etc. Collaborative applications need to support dynamic groups that can scale to large numbers of users. A peer-to-peer model inherently makes these applications easier to design and to operate for groups. Since there are no servers, groups can form ad hoc and there is no setup or scheduling with a centralized authority required.

1.1 What is peer to peer ?

It is network of peers (workstations) in which each peer is having an equal responsibility. So in a way they differ from client server architecture where where communication is usually to and from a central server. P2P relies on peers to take active part in management of network communication. There is a growing demand of group-oriented applications over internet Applications.

1.2 Scenario

Distributed environments where peer interact and collaborate in an dynamic yet self organizing manner requires a fine grained access and admission control framework. We will look one of the scenario for which we have implemented a base framework.

Secure Messaging: Consider a chat room scenario, in absence of a central server which acts as an admission authority for the group we require that authorization decisions are

being taken collaboratively by the members of the chat group to allow new member. Access rights of a user may vary like some may only be allowed to receive the messages or message is being received by restricted people in the group so we may need to assign roles to peers in the group which we have modeled as different hierarchies of the group. Now access rules are a function of hierarchy to which a peer belongs.

1.3 Why peer to peer ?

An important goal in peer-to-peer networks is that all clients provide resources, including bandwidth, storage space, and computing power. This is not case for of a client-server architecture with a fixed set of servers, in which adding more clients could mean slower data transfer for all users. Peer to peer groups have decentralized control thus they avoid single point failure. Due to decentralized nature they can resist to intentional DoS (Denial-of-Service) attacks. Server-client architecture can be in trouble if servers are down due to load caused by large number of requests at same time. When as large number of clients are connecting to same server, we need high performance machines at server end, thus is more costlier as compared to P2P architecture [4].

Peer groups provides us flexibility P2P architecture but we also require that framework be so that it is adaptable to dynamic number of peers and also does traffic load balancing. So in P2P scenario each peer is communicating to the other but requirement is such that number of message exchanges should be minimum. Along with all these feature, for a secure group communication we will require that re-keying operation should be done when user joins or leaves in order to maintain forward and backward secrecy.

1.4 Related Work

Several attempts to tackle security issues in collaborative groups have been made which includes Antigone project, Secure group layer SGL etc. We have studied different models of access control that have been applied for group communication. Some of than can be listed as:

- Identity Based Access Control (IBAC)
- Role Based Access control (RBAC)
- Authenticated Role based access control (ARBAC)
- Attribute based access roles (ABAC)
- Trust based access control.(TBAC)

We have discussed some of them thoroughly later. In distributed system peers do not know each other, thus for functioning of group trust and reputation mechanism come into picture. Such systems establish trust among members by using feedback from peers. Example of such system includes eBay, an online auction site where peer rate each other after every transaction. However this system relies on a central server for storing and managing ratings. Other such example is Kazaa which is a file sharing system that provides a reputation system based on Integrity rating and participation level.

1.5 Overview

We have proposed an adaptive access control framework which integrates role based access control, policy based access control and whole of distributed system running on a trust element among peers. In this report we discuss different aspects of admission and access control where we sketch steps involved in admission of new peers to an existing group. In chapter 3 we present important aspects and need of key management. Chapter 4 gives a brief overview of JXTA (peer to peer architecture), which is base for our implementation and we also discuss about our implemented design along with presenting possible future work in this area.

Chapter 2

Admission and access Control

In Peer to Peer group scenario we will require that there are some standards that defines which user can join the group and once they are granted admission we need some policy to regulate what resources the peers can access. So we need to define Admission and access control polices. Here terms can be confusing but to get a clear view, admission control is till joining of new peer to the existing group whereas access control policies play important role in secure access to group messages and resources. This section presents a description of what is admission and access control.

2.1 Admission Control

In Dynamic Peer groups where every user is free to join the group, if anyone is able to obtain the access to the group key, group key management becomes useless. Therefore, mechanisms to control membership are required. Hence along with key management, peer groups should have some policy on how to decide upon whether to allow a new user to enter the group. Admission control (membership control) is needed to allow only authorized users to join the group. This is a very important issue, since all other security services rely upon group membership. For a peer group admission can be done in several ways but before that prospective group members must be able to get a clear idea about group and how to gain admission (or what the criterion's are.) So group information and admission rules must be well electronically documented. Ways in which admission control [5] can be achieved via:

- **Admission via ACLs** : All group members are known before group creation. So if peer is listed in ACL then he will be able to communicate with others.
- **Admission by Group Authority** Decisions regarding allowing new user to group are taken my group authority who may be the group creator or some elected member. We will require that Group authority must be trusted by all group members. But here Group authority need to be online all the time along with it must be resistant to attacks.
- **Admission by Members** For admission of a new member some or all existing group member take part in decision of admitting new member. One example can be when a new request arrive for group joining. All members perform an voting whether to allow new member or not. If the vote count is greater than some threshold then new member is eligible and an certificate or token is issued to him. But group

based admission where all take part in decision may be complicated to implement as compared to other methods.

In our implementation we have focused on making system completely distributed and it is responsibility of all for functioning of group so it is required that decisions like admission to group are taken by considering opinion of every member. So in our case peer, when a join request comes a voting procedure is initiated and if percentage of votes in favor is above some threshold according to policy the admission certificate is granted. In next section we will elaborate important steps in admission procedure.

2.2 Admission Process

A new peer would pass through these phases when following admission process for a group.

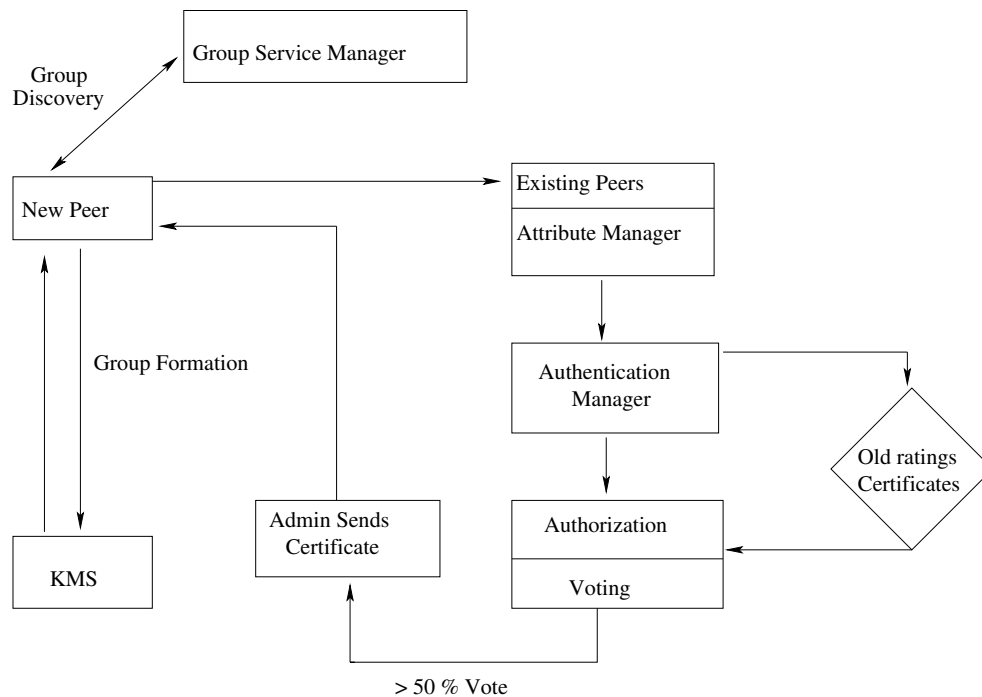


Figure 2.1: New Member Join

Here are the basic step which are common to all schemes where admission is done by collaborative efforts [1] .

2.2.1 Advertising the created groups

One way of getting information about existing peer groups can be to get the advertisements via some special peer which works like a public website (central server) which keep information what are currently existing peer group and also how to gain access to them. Other way can be that new peer group flood a query into the system and get response from other peers which detects that some new peer want to join group and thus sends him a response which include group charter(Group charter is electronic document which

holds information about group and its peers. Now the new member has got the group charter, it contains various parameters and admission policies, including: group name, signature/encryption algorithm identifiers and other optional fields. This process is performed only once per admission.

In our implementation, to make it fully distributed and independent of a central server, it is required that initiator of group should publish and advertise his group along with group charter. Also the group advertisement is being periodically broadcasted by other members of group once they are part of it.

2.2.2 Request for joining group

A new peer wishing to join first searches for interested group among different group advertisements available in the network. Then he would request any member of the group to consider his joining request. His request includes his identity, public key along with requested hierarchy in the group.

A peer credential could be created by hashing the concatenation of unique user id UUID which in our case is peer name and public key fields and then signing the hash with the private key of the user and using this digital signature as identity of the peer. This identity could be used as peer's credential for the messages.

$$P_{new} \longrightarrow P_i : \{JoinRequest\}_{PrivateKey}, Cert$$

$$Cert = UUID, PK, Rating, \{H[UUID||PK]\}_{PrivateKey}$$

2.2.3 Authentication of user who wishes to join

The peer receiving this signed request will obtain the identity of requested peer from certificate. He will compute the hash of UUID and PK and tally this with received message. If the two hashes match then the user is authenticated and will be granted access based on access policy. And from now own a binding between user and his public key is made. If the receiving peer is not aware of any previous rating certificate of applying peer then voting process is invoked by broadcasting this join request to other members of the group. Also this join request is encrypted with private key of broadcasting peer so that it can be verified that request is forward by a known member of the group.

2.2.4 Voting

Collaborative decision by majority peers as to whether to allow new peer to join or not. Possible approaches can be:

- Existing members may decide based on majority of votes whether to allow new member.
- Static ACLs (access control lists)
- Admission by some of authoritative servers.

In our implementation we have chosen the first approach. All peers receiving the broadcasted join request verifies the identity of peer initiating this request. Peers return the result of voting to Access Policy Manager of the peer who initiated it.

$$P_n \longrightarrow P_i : \{Vote, Level\}_{PrivateKey_n}$$

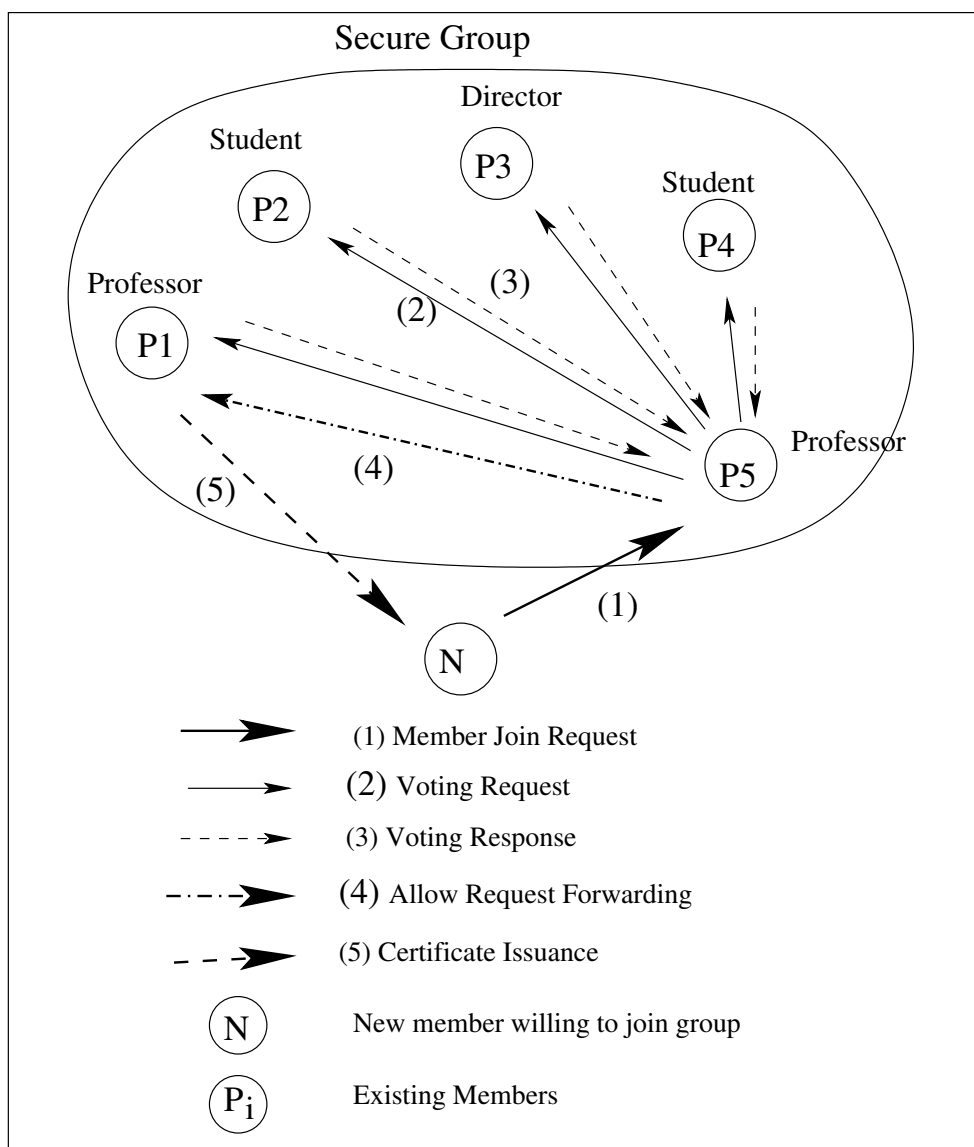


Figure 2.2: Admission process

2.2.5 Issuance of Admission certificate

Once enough votes are collected, Access Policy Manager verifies all the votes, and decide whether to accept the new node (N) as a member based on Admission policy. If the requester is qualified, Admin sends an X509 authentication certificate to N. N plugs this X509 certificate into the membership service of group. The membership service verifies whether this certificate is issued by Group Admin only. On successful verification N can listen and send messages within the new group joined.

2.2.6 Group Rekeying

If the new peer N has joined at Hierarchical level(H_i), rekeying is done for that level. Having the new group key, member communicate using DES encryption with key K l corresponding to their level. This completes the join process.

2.3 Access Control

For any group we need to state that what type of resources are accessible by a particular peer. In a way access control policies define set of rules to decide upon what all resources are accessible to a group member under what conditions.

There can be two types of scenarios, Static and Dynamic Groups. In former case information about all group members is known in advance. So here management of access policies is pretty easy and can be done using ACLs (Access control lists, ACL is a table that specifies which access rights each user has to a particular system resource and service). But in later case the peers can join and leave at any time so it may be difficult to implement polices for dynamic groups. Access control can be divided to two categories :

- **Role based access control (RBAC)** : With role-based access control, access decisions are based on the roles that individual users have as part of an organization. When a new member join the group he/she will get a list of available roles that the application(group) supports. The new member will request for role (may be highest available) passing his identity to other peers. If new member's admission request is rejected, he can attempt to join in lower role. Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to have associated role. So here permissions are associated with roles not with individual peer [3]. Now we can divide group based on roles. It can also help us to have weighted decision is some cases. (fig2.1)

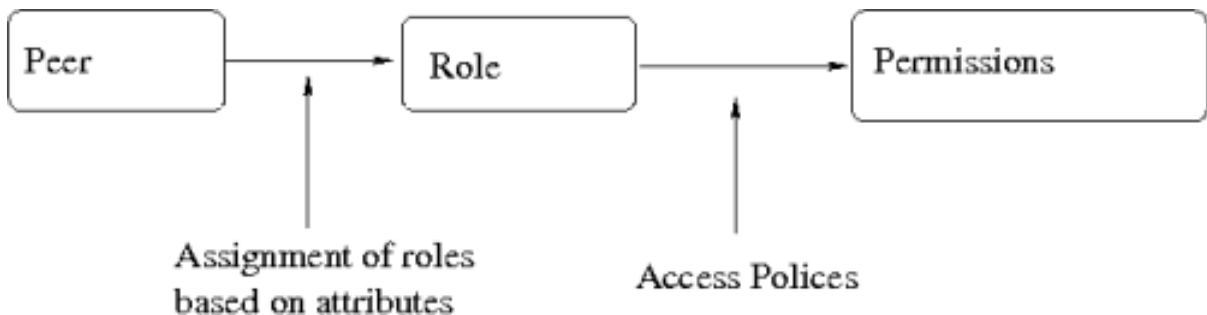


Figure 2.3: Role Based Access Control

- **Attribute based access control (ABAC)** : In this case access permission are determined based on authenticated attributes of user. Attributes are normally function of user's identity and its characteristic. So a policy rule which decide whether a peer p can access some resource r is a function of peer and resource attributes. So a rule will look like :

$$RuleX : can_access(s, r) \leftarrow f(ATTR(s), ATTR(r)) \quad (2.1)$$

But some times identity based schemes do not provide enough flexibility like when group size is very large, In such cases we will need that group is divided in different hierarchies which can be based on roles of peers. But sometime it has its own advantages as compared to Role based access control(RBAC). In case of RBAC, fine grained access control polices often involve multiple subject and attributes. As the

number of attributes involved increases, number of roles and permission needed to encode these attribute increases exponentially. RBAC might need some centralized server to manage user to role and role to permission assignment.

Access Control is achieved by maintaining different keys (K_i) for different hierarchical level. On successful join by new member N, group rekeying is done corresponding to the N's hierarchical level (H_i). For ex: If a new Professor wishes to join at his level then rekeying is done among all the Professors. This makes the communication between members of level H_i secure from all other hierarchies. This is because all the members from level H_i communicate among themselves by encrypting messages with K_i so that members belonging to that hierarchy only can decrypt message. This is an example of *Role Based Access Control*.

With respect to our implementation, we have just provided a basic Access control policy but there is much scope that need to be covered but our implementation proves our purpose of demand of Access control policy and its proof of possibility of implementation.

2.4 Member Level Updation

Once a peer is member of the group to enjoy more access privileges he may periodically request group members to rate him. It can also be though as performance evaluation and trust gained by a peer in the system as rated by others.

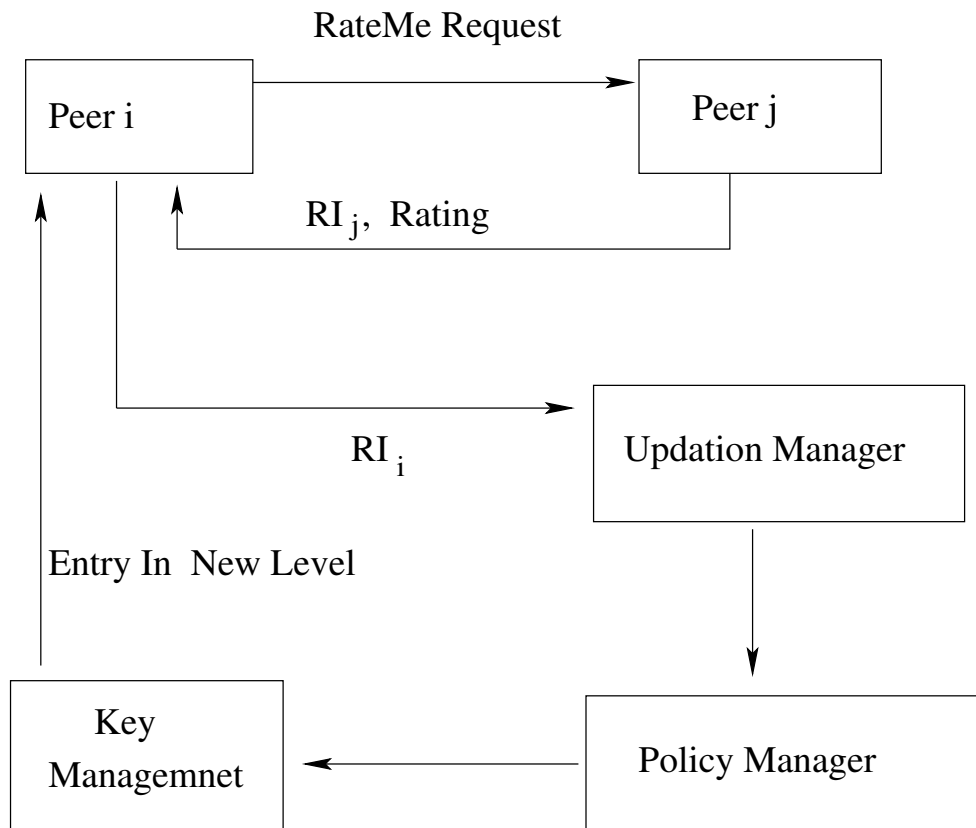


Figure 2.4: Member Level Updation

Consider a multiplayer gaming scenario, where a group consists of people who are interested in that game. As each game play results in some decision like winning or losing. Thus peers can be rated based on their performance in game or any other criterion. So a player with a higher ranking can be allowed to play in higher levels. Also for higher level the access rules may differ from others.

In our implementation we have created a peer to peer chat client where peers belonging to same group/hierarchy can chat. In order to get level upgrade a peer may initiate a rating request which is being sent to other peers in order to rate him. Now other peers rate him on a scale of 10 (as in our case). And the recommending peer sends him a certificate which comprises of following fields:

- Recommending peer's identity
- Recommended peer's identity
- Recommending peer's own rating
- Rating given by recommending peer
- Signature of recommending peer.

Other fields that can be included to tackle security issues can be: Issuing date and time, Expiry date and time. After receiving rating certificates, The trust Engine calculates new rating of the peer and it is being passed to Updation manager which interacts with Policy Manager to update the level of peer. See Figure 2.4

Thus reputation index(i.e rating) of a peer i is calculated as weighted mean of recommending peers rating with recommending peer's own rating. as weights.[8]

$$R_i = (R_j * T_{ij}) / \sum T_{ij} \quad (2.2)$$

where T_{ij} is the rating given by peer j to peer i.

Chapter 3

Key Management

Group key management protocol used must be secure, efficient and meet the application needs. Security concern is that group key should not be easily known or deducible by an outsider. Group key management protocol must be scalable according to application needs.

Each group requires a secret key which is used to encrypt all messages between different peers to in order to maintain secrecy of communication. So to send a message, the source encrypts it with group key and send it. Upon receiving the encrypted message, only an authentic member will know the secret symmetric key thus only he can decrypt the message.

Security requirements in key management [2] :

- *Forward Secrecy* requires that users who have left the group should not have access to any future key. This ensure that a member cannot decrypt data after it leaves the group. So when a member leaves the group a new group key should be generated through re-keying. To summarize re keying will allow new member to decrypt future message but not previous messages.
- *Backward Secrecy* It requires that when a new user joins so he should not have access to previous group key as he should not be able to access messages prior to his joining. So a re-keying operation is needed each time a new group member joins.
- *Collision Freedom* Current session key should not be deducible by any outsider.
- *Key Independence* Disclosure of a key should not compromise other keys.
- *Minimal delay* Group key should be computed in minimal amount of time thus minimizing delay in packet delivery which is sentive for multimedia applications.

We can see that as a new member joins or previous member leave we will need to compute a group key again each time in order to maintain secrecy of group messages. So a single membership change is affecting all other group members. So we will need a protocol which does least number of message exchanges between peers to form a new group key in order to have lower bandwidth load.

3.1 Categorization of Key management protocols

- **Centralized** : A central authority distributes group keys to group members. This central authority can be Trusted third party or can be group authority.

- Decentralized architectures : The management of a large group is divided among subgroup managers, trying to minimize the problem of concentrating the work in a single place i.e. avoiding single point of failure.
- Distributed : The group key is agreed upon by all group members uniform contribution. For example we can do a N-party Diffie Hellman key exchange.

3.2 Key management Protocols

In case of peer to peer topology, group member cooperate to establish a group key. Here we will discuss some of the group key management protocols [2] used in distributed systems.

- Ring Based Approach (using Group Diffie Hellman) : In this protocol members are organized into a virtual ring, so member M_i communicate with member M_{i+1} and member M_n with M_1 . Group computes the key with n-1 rounds. Here each member comes up with a random number N_i .
- Hierarchy based cooperation : Skinny Tree (STR) (Fig 3.1) STR is distributed/contributory protocol which is primarily variations of the n-party Diffie- Hellman key exchange. Here members of a group are organized at leaves of tree. Each leaf hold its secret key and it calculates g^{k_i} and propagates towards its parent. now a combined key from children's g^{k_i} and $g^{k_{i+1}}$ is calculated so at end group key is calculated at root. In case of membership change (join/leave) the tree is r-built consequently and hence all the member update the group key which is the new key associated with root of tree. The protocol relies on the fact that every member can compute group key if it knows all blinded keys in the key tree.

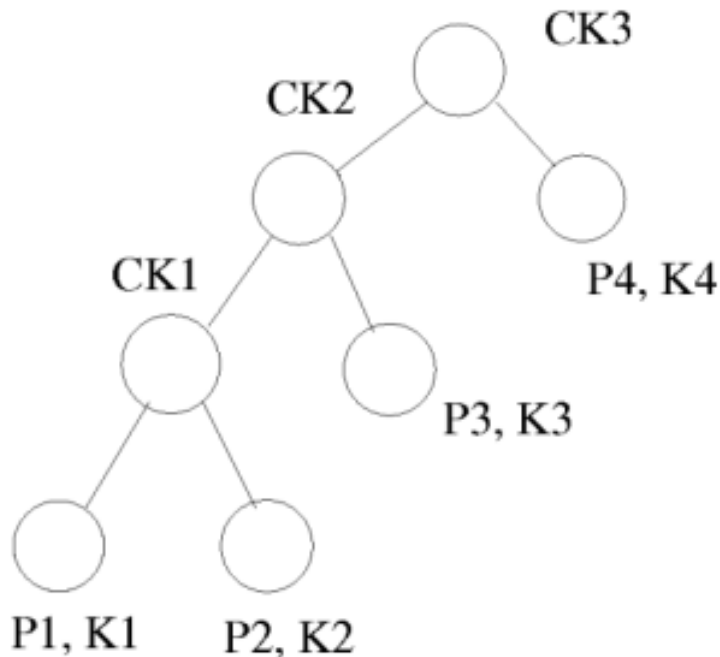


Figure 3.1: Skinny tree protocol

The final computed key will be:

$$g^{k_n * g^{k_{n-1}} \dots g^{k_2} * g^{k_1}}$$

Member Join When a new member M_{n+1} wishes to join:

1. M_{n+1} broadcasts its contribution ($g^{k_{n+1}}$) to all other members.
2. Each member then recomputes the key.
3. Right Most member of the previous tree (M_n) act as group controller and send contributions ($g^{k_1}, g^{k_2} \dots g^{k_n}$) of all previous members to M_{n+1} who then computes the new key

Member Leave When a member M_i leaves, renumbering of nodes is done. Now M_{n-1} selects new random key k_{i-1} and calculates all CK_i 's and broadcasts this to all members thus everybody is able to calculate key now.

- Broadcast based approach : This approach relies on broadcasting secret messages and calculating group secret key which is function of secrets sent by all members. One example can be using Diffie-Hellman, then peer p_i comes up with a random number r_i and now it can send this to other peers by encrypting it with previous group key. So each sends its secret to every other peer. Now each peer can calculate group key, $K = g^{r_1 * r_2 * \dots * r_n} \text{ mod } p$, where p is a prime number and g is primitive *mod* p .

In our implementation we have used STR protocol for Key Management. And for each hierarchy one such tree is maintained thus each hierarchy having a different group key.

Chapter 4

Implementation of Secure MultiChat

We have implemented a secure chat application based on JXTA where peers interact in completely distributed fashion. Each group could have different class of members. Considering IIT like scenario various users may join a session in different roles e.g professor, student, and Director. Users are authenticated based on static access policy and voting. After authentication and voting, group rekeying is done and entry of new peer is broadcasted to all the members of the group. Peers are also capable of updating their level by collecting feedback from other peers. For key management we have currently used STR protocol which uses Diffie Hellman algorithm to achieve rekeying. In this chapter initially we present a introduction to JXTA and then will discuss our implementation.

4.1 JXTA: Peer to Peer framework

This section provides a brief overview of JXTA framework. JXTA technology is a set of open protocols that allow any connected device on the network. The JXTA protocols standardize the manner in which peers:

- Discover each other
- Self-organize into peer groups
- Advertise and discover network services
- Communicate with each other
- Monitor each other

So one can build up application on top of these protocols. The main components in a JXTA network are : Peers and peer groups, Messages, Pipes, Services and applications, Advertisements.

A peer group forms a boundary that prevents non authorized users from accessing to group
JXTA divides software to three layers :

- JXTA core - handles peer discovery and monitoring
- JXTA services - Generic services like files sharing and indexing
- JXTA application layer is reserved by application developed by JXTA community.

In case of JXTA peers are organized into peer group which is an important functionality of JXTA. Peers communicate by sending messages over JXTA pipes. Because pipes make use of underlying transport protocol so nothing can be assured about reliability of messages sent over JXTA pipes. But we can design our framework so as to ensure reliability by implementing our own scheme.

- JXTA uses a small number of protocols. Each is easy to implement and integrate into P2P services and applications. Thus service offerings from one vendor can be used transparently by the user community of another vendor's system.
- JXTA is independent of programming languages, so that it can be implemented in C/C++, the Java programming language, Perl, or other languages. Heterogeneous devices with completely different software stacks can interoperate with the JXTA protocols.
- JXTA is independent of transport protocols. It can be implemented on top of TCP/IP, HTTP. This means that a system built on top of JXTA functions in the same fashion when the system is expanded to a new networking environment or to a new class of devices, as long as there is a correct transport protocol handler for the new networking protocol.

Using JXTA as set of underlying protocols we have created an secure messaging application which works in distributed manner as opposed to client-server architecture. Its functioning is governed by set of rules which defines group's admission and access control policy. Next section will take through an overview of our implemented design.

4.2 Implementation

We have implemented a peer to peer model where a communicating group is defined as set of peers having unique identity associated with their public-private key pair. Groups are formed based on particular interest criterion. Group membership is governed by set of rules which defines the admission control policy of the group. Our implementation supports joining of group based on voting done by pre-existing members of group for a new member. Dynamics of group are such that peers do not need to be online all the time, peer may leave and join at any time.

In distributed systems involves multiple peers which often don't know each other. So there is risk factor involves whom to allow to join the group and access resources. This has given rise to trust and reputation mechanisms in peer-to peer systems. So a peer can be rated based on his behavior and can provide a measure of how much a peer can be trusted for a transaction.

We have analyzed two possible approaches and have developed test applications for both.

1. *Central Server for Authentication:* When a peer instance is initiated it registers itself to central server by proving its possession of public key which is then stored on central server. Central server acts as a trusted third party and gives new peer a X.509 certificate signed by him. Now for verifying his identify, other peer contacts central server to access his public key and then sending message encrypted with public key of peer. In response that peer can decrypt it with his own private key hence a secure medium of message exchange is established.

2. **Trust Based Peer to Peer communication:** Here system works in purely distributed manner as compared to server-client architecture, group is responsibility of all the peers. In such scenario when a instance is initiated (and after joining a group) it broadcasts its public key to all other members of the group. In such a manner all peer have public key of every other peer. Once other peer gets his public key, his identity is bound to that public key. Now anybody pretending like him will not be able to decrypt the messages as they can only be decrypted using given peer's private key. So here a peer do not need to contact central server for accessing each other's public key but there is very small overhead of storing public key and additional information about peers is involved. Here Peer will need to provide a self signed pseudo certificate with proof of possession of private key.

After successfully joining the group, peer membership can be controlled by feedback mechanism. Currently our implementation is such that peer himself can only initiate his level updation request but it is easy to extend it such that other peers can initiate level updation for fellow group members. Such that peers falling below some threshold may be kicked out of the group.

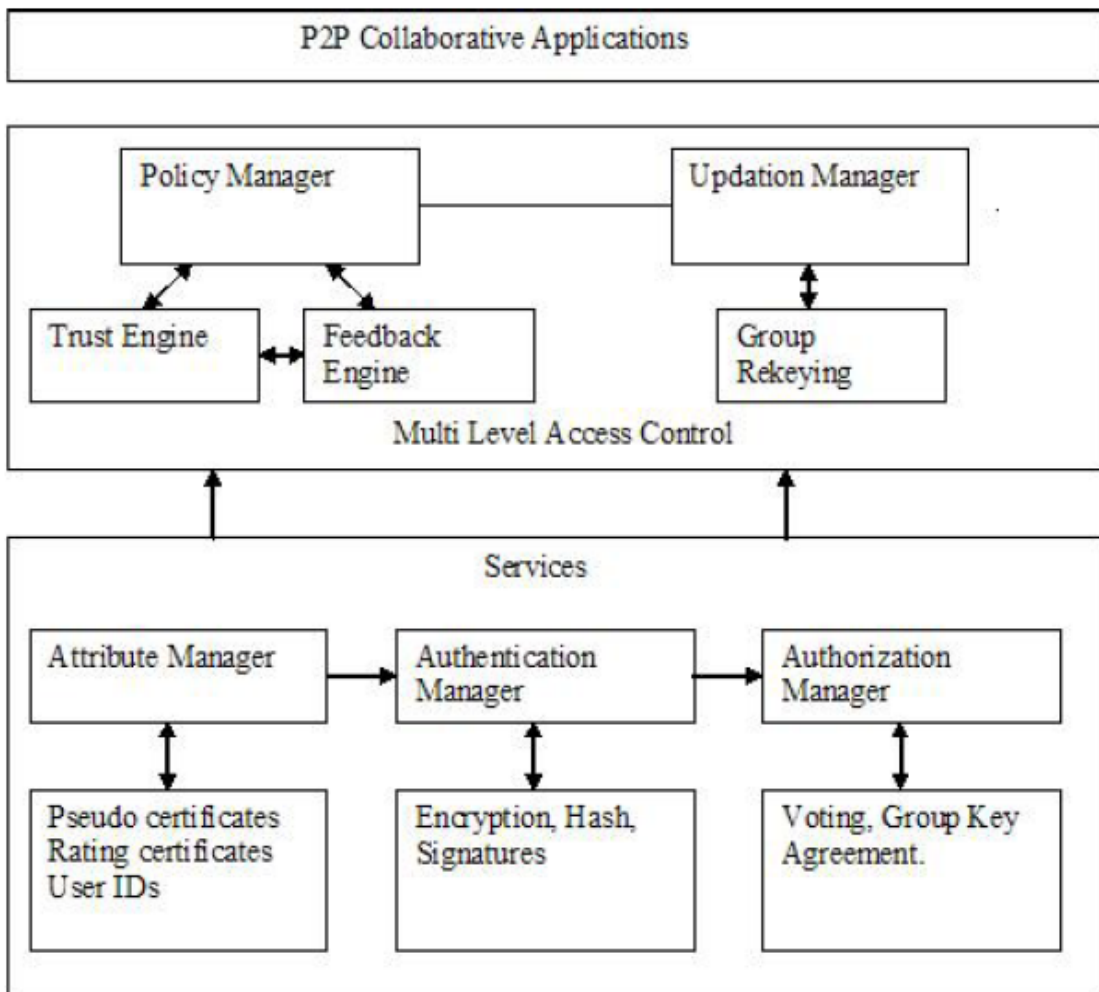


Figure 4.1: Integrated Framework

4.3 Functional Components of Framework

In a distributed system the responsibility is distributed over different peers of group. Though each individual peer is capable of performing the task of authentication, admission and access control and key management but the privileges are restricted by peer's role in the group. So important functional components a peer can have are:

- **Attribute Manager:** Stores the attributes and public keys of the group members.
- **Authentication Manager:** Verifies the validity of identity and invokes the voting process.
- **Feedback Manager:** Calculating rating (reputation based on feedback from other peers in form of rating which is based on his performance in the group).
- **Key Management System:** Invokes rekeying process when a new peer joins or any existing peer leaves.
- **Access Policy manager:** Frames rules based on attributes, behavior of peers. Checks the rules to allow members to join.
- **Updation Manager:** Responsible for level updation based on reputation index of peers after they have collected feedback form other peers. Also responsible for granting or revoking access rights to peers based on policy.

4.4 JXTA Messages

Components belonging to different peers interact with each other using two type of communication channel that are being supported y JXTA:

- **Broadcasting:** Not reliable, message is being broadcast to everybody in the group. In our case peer advertisements, chat messages, join and leave notification are being sent in broadcast fashion. As everybody receives these message they are encrypted with group key, so only group members can decrypt it.
- **Unicast messages:** This requires pipe address of peer whom we need to send message and then message exchange is pipe to pipe. Here for security reasons we encrypt the message with pubic key of peer to whom we are sending message. Voting and rating messages are example of unicasts messages.

In our implementation we have used both channel for sending various messages. In order to provide appropriate response to such messages in our implementation two eventlister threads are running:

- One responsible for handling broadcast messages, analyzing them and doing appropriate action and then sending response accordingly.
- Other responsible for listening to unicasts messages.

4.4.1 Unicast Messages

Each Unicast message has a description field to uniquely identify the message. Based on description field different types of Unicast messages are being handled by our implemented client:

- **Request Join Group:** When a new instance of application is being created and peer finds a group advertisement then "Request Join Group" message is being sent to member of a group indicating that he is interested in joining the group. So peer receiving the request verifies sender's identity and requested hierarchy of aspirant member. After that it sends as "Voting Request" to other peers and then collect vote and based on that decides the Denied and Granted status of "Join Request"
- **Request Voting:** Message sent by Election thread of application which involves sending voting requests to all existing members of the group.
- **Request Join Group Granted Forward:** Message sent to Admin to grant the new member the certificate for joining the group.
- **RateMe:** Request sent by a peer to other members requesting feedback from others. And then reputation index is being calculated from received feedback. Based on reputation index the level updation of peer takes place.
- **UserPublicKey:** Sending of public key to all other peers.
- **OldKeys:** Signaling the Key recomputation when a new peer joins the group.

4.4.2 Broadcast Messages

Different type of Broadcast messages:

- **Chat Messages:** All such messages are being sent through broadcast medium. But to maintain security messages are encrypted with group key so that only legitimate group members should have access to group messages.
- **Join and leave notification**
- **PostLeave:** When a peer receives the leave notification of a peer then it invokes a key recomputation in order to maintain forward secrecy.

Other than these messages a peer instance periodically sends messages to other peers which includes:

1. Peers broadcasting their identities (Public Key, ID, hierarchy)
2. Publishing their group advertisement.

4.5 Policy Language

To represent rules which governs admission and access control of peers a policy language is required. We developed a small policy language which standardize how rules can be defined. We have used XML to specify set of rules for policy language framework, as it provides a customizable and extensible framework. It is also easy to send XML messages in JXTA. In fact JXTA's group and peer advertisements are in form of XML files. Many

programming languages in our case Java has inbuilt support for reading and writing XML files, also providing support for easy searching and validation. XML can also be used as specification language for describing peers, peer groups and services. A sample rule file:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rules>
    <request type="join">
      <min parameter="votecount" type="percentage">50</min>
    </request>
    <request type="update">
      <newlevel id="1">
        <min parameter="rating" type="number">6</min>
      </newlevel>
      <newlevel id="2">
        <min parameter="rating" type="number">8</min>
      </newlevel>
    </request>
  </rules>
```

Figure 4.2: A sample rule file

Above policy language presented in fig 4.2 specifies two types of rules which defines admission and updation of peer. First rules stated that peer can be granted admission to group if votes in his favor are more than 50%. Second rule specifies minimum rating level required for getting upgraded to different levels like for getting privileges of level 2, it is required that rating is above 8.

At present our implementation support rule file specifying limited set of rules but it can be extended to support wide variety of rules. So present structure can be consider as a proof of concept that a dynamic policy based framework is possible and can implemented using JXTA. But still attempts need to be made to standardize other kind of access rules and making it dynamic such that peers can collectively modify the policy.

Chapter 5

Conclusion

To summarize, this project aimed at providing an integrated framework and a testbed for secure communication control which provides :

- Admission control based on authentication of peers.
- Hierarchical Access control (Role based, with trust as one of attributes).
- Re-keying which involves key distribution in order to maintain forward and backward secrecy.

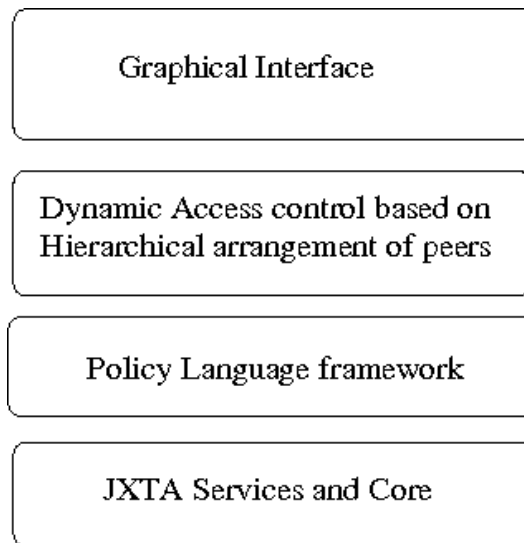


Figure 5.1: Framework for Secure Communication control

We presented a framework for which we successfully implemented a secure chat application which uses admission and access control along with key management using JXTA (peer to peer architecture) as our base. We examined the basic steps involved in group creation and peer admission. Also member level updation using feedback mechanism was successfully incorporated in our design. We also studied how key management can be used to provide forward and backward secrecy along with preventing attack from external entity.

5.1 Analysis

This section presents a picture of the contrast between central server based approach and one fully peer to peer in terms of message exchange. Number of message exchanges in former case (Let no of peers in group = n) :

- **1st message** sent by new peer (P_{n+1}) to one of peer (P_i) of group sending join request.
- **2 messages** sent for verification of P_{n+1} by P_i using central server.
- **2 + 1 message** exchanges involved in forwarding join request to other peers for voting after verifying its identity.
- **2 messages** for verifying identity of P_i
- **1 messages** for voting response.
- **2 messages** exchanges for forwarding voting result to Admin and then he issues certificate to new peer.

Time delay between applying for a group and getting certificate for that group is of **11 message** exchanges. While in fully peer to peer architecture, authentication is done by stored public key. So only **5 message** exchanges are required which is at a cost of very small space overhead.

5.2 Future Scope

Our implementation is a starting point toward a dynamic framework based on policy language defining admission and access control. Our implementation provides us a proof of concept and a testbed for an framework that enables secure group communication in a distributed manner, . In future, work need to be done in these areas:

1. Implementing multiple join-leave. (Concept of batch rekeying)
2. Making policy language dynamic and open to wide variety of rules.
3. Dynamic leaving of members: Situation can be handled easily if peer leaves after successfully informing other peers. But to tackle situation where peer leaves without informing, or program terminates accidently or other failure. Solution can be checking periodic peer advertisements. Periodic rekeying could be another solution to tackle security threat from such peers.
4. In Multiplayer gaming and file sharing centered peer groups we can include the transaction cost as one of factors affecting a peer's credibility. Like the game result in multiplayer case may be criterion for his performance.

Bibliography

- [1] Yu Zhang, Xianxian Li, J. Huai, Y. Liu : *Access control in Peer to Peer collaborative Systems*, ICDCSW'05, IEEE-2005
- [2] Y. Challal, H. Seba : *Group key management protocols: A novel taxonomy* IJIT 2005
- [3] J. F. da Silva, L. P. Gaspar, M. P. Barcellos and A. Destsch : *Policy-based access control in peer to peer Grid systems* Grid Computing Workshop 2005 , IEEE-2005
- [4] A. El Saddik, A. Dufour: *Peer to Peer suitability for collaborative multiplayer games* Proceedings of the Seventh IEEE International Symposium on Distributed Simulation and Real-Time Applications, 2005
- [5] Yongdae Kim, D. Mazzocchi, G. Tsudik: *Admission control in peer groups* Proceedings of the Second IEEE International Symposium on Network Computing and Applications, p.131, April 16-18, 2003
- [6] Y. Sun, K. J. Ray Liu: *Scalable hierarchical access control in secure group communications* INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, March-2004
- [7] H. Tran, M. Hitchens, V. Vardhanaraj, P. Watters: *A trust based access control for P2P file sharing systems*, Proceedings of the 38th Hawaii International Conference on System Science, 2005
- [8] Sepandar D. Kumar, Mario T. Schlosser, Hector GarciaMolina: *The EigenTrust Algorithm for Reputation Management in P2P Networks* WWW2003, May 2003, Budapest, Hungary. ACM
- [9] "Sun Microsystems Project JXTA v2.3.x:Java: Programmer's Guide"
<http://www.jxta.org/>,2005

Chapter 6

Appendix

6.1 Screenshots of our Chat Application

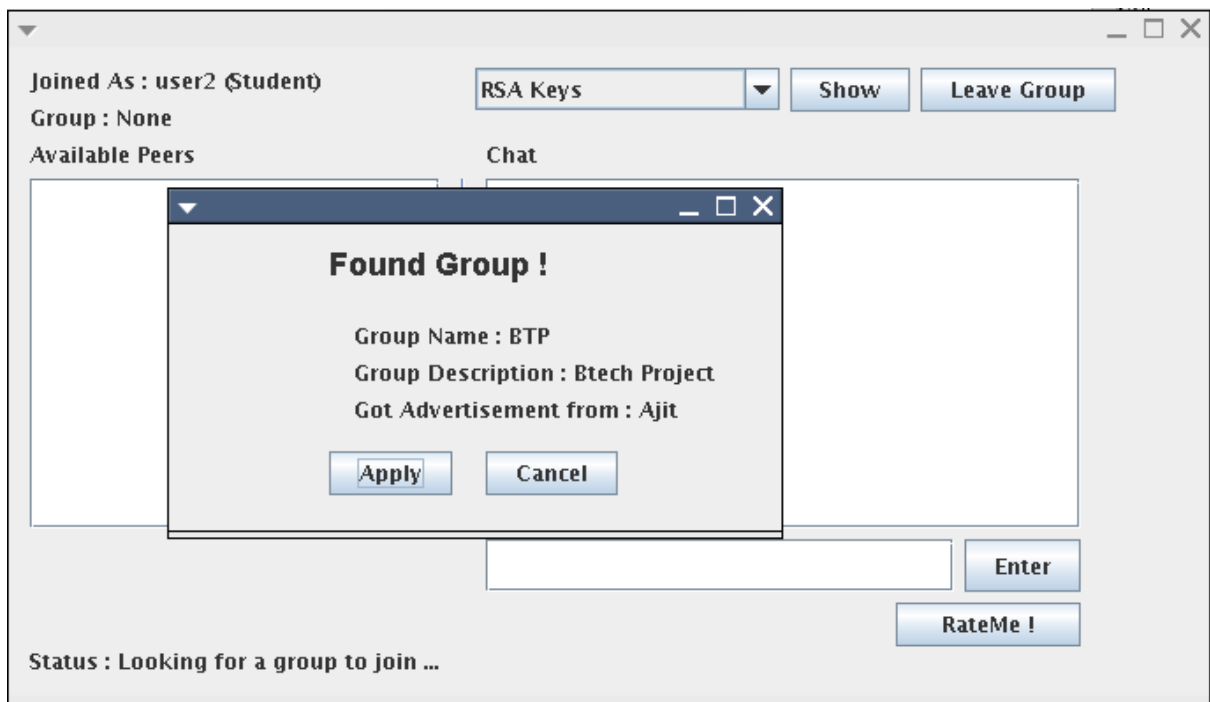


Figure 6.1: Found a group advertisement

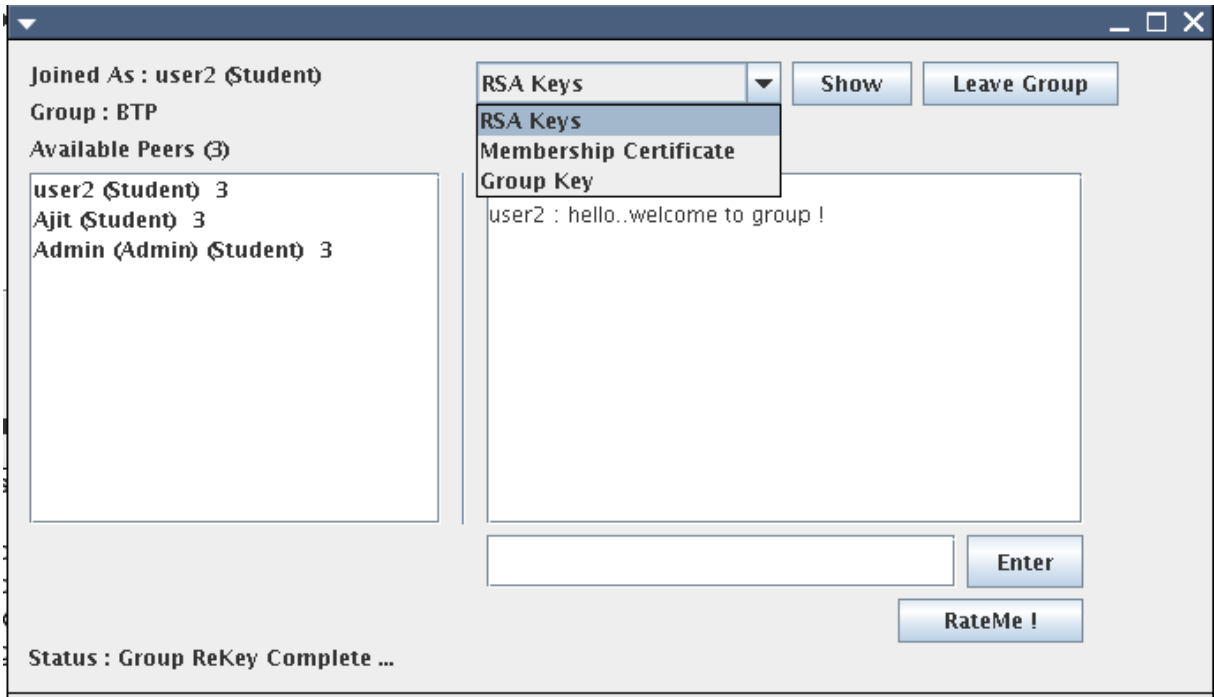


Figure 6.2: Chat client

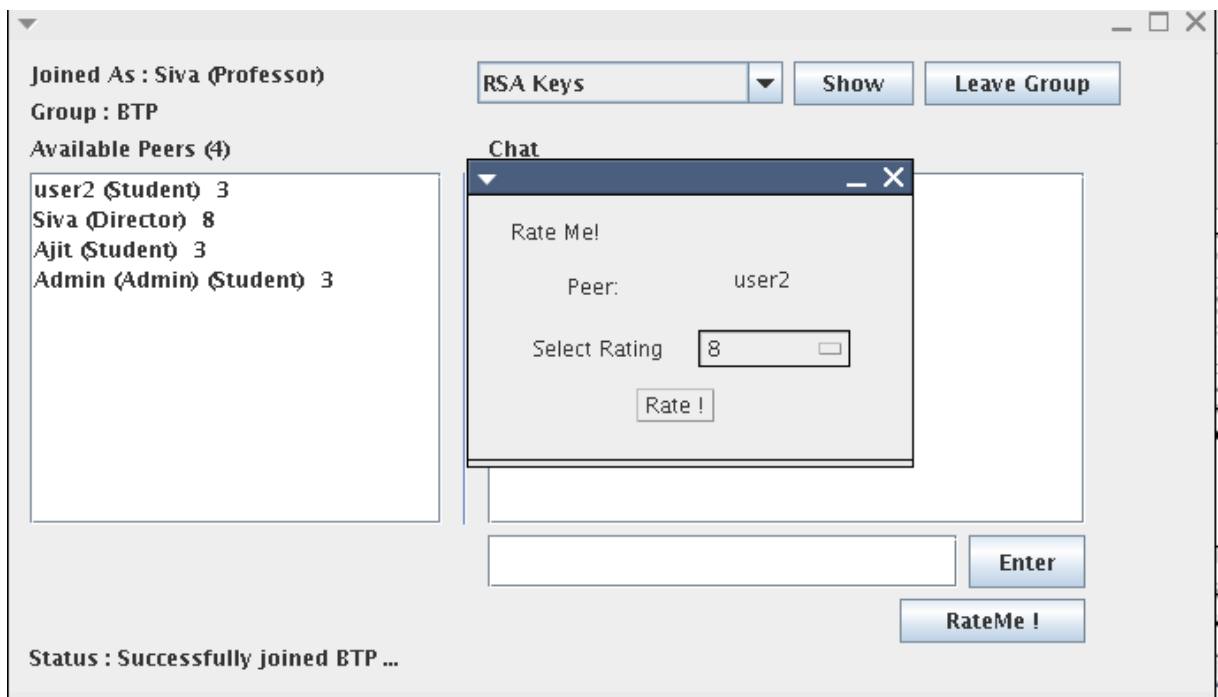


Figure 6.3: Rating request

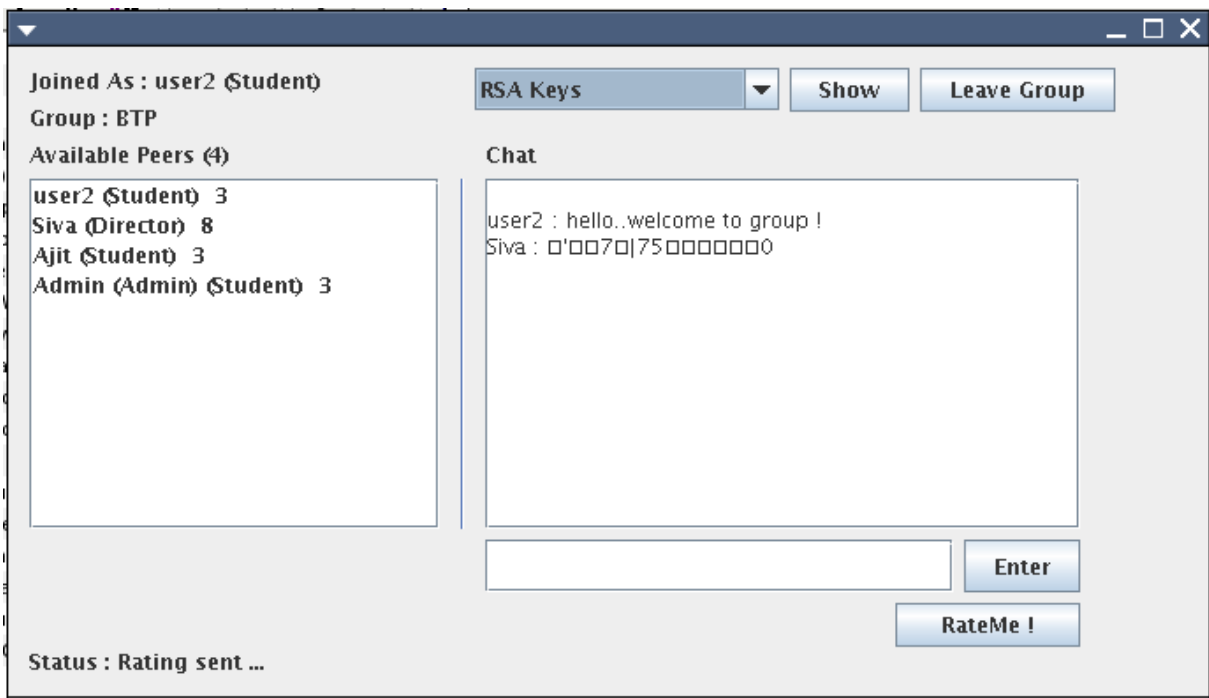


Figure 6.4: Rating Updated

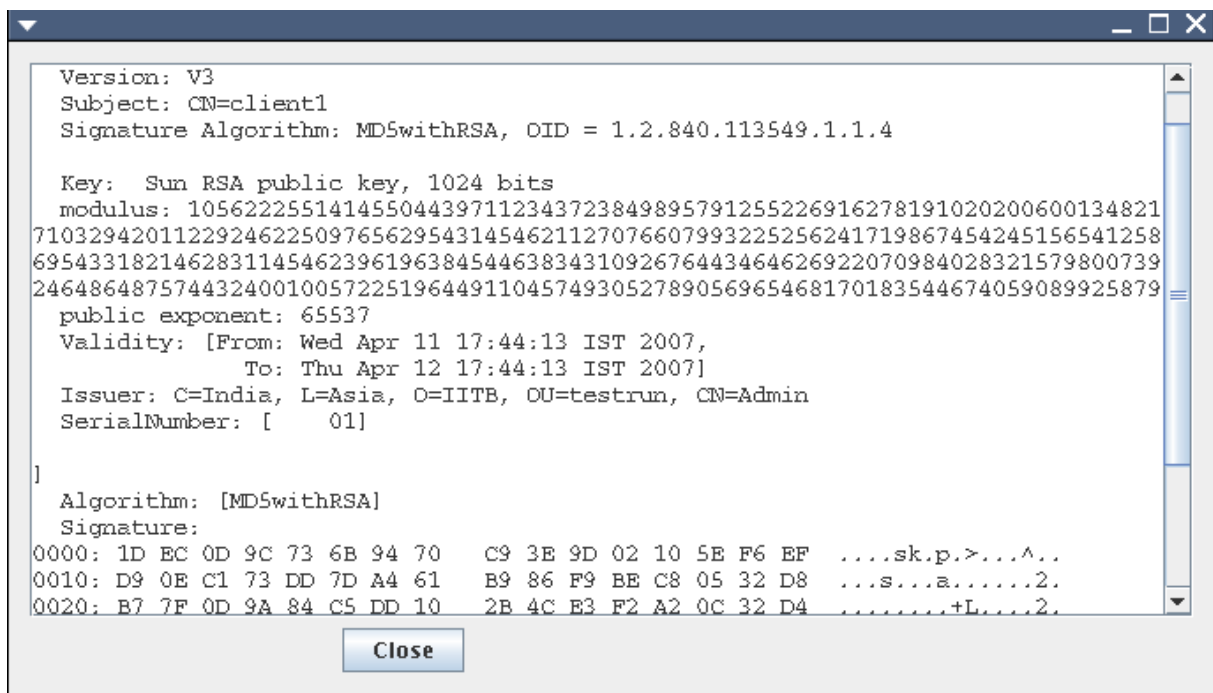


Figure 6.5: Certificate