Integrated Framework for Authentication and Access Control in Peer to Peer Groups

Madhumita Chatterjee, G. Sivakumar, and Bernard Menezes
Indian Institute of Technology, Mumbai, India
madhumita@cse.iitb.ac.in, siva@cse.iitb.ac.in, bernard@cse.iitb.ac.in

Abstract. Collaborative applications like video/audio conferencing, IP telephony, file sharing, collaborative work spaces, and multi-user games, having varied security requirements, require a secure and reliable group communication system to provide co-ordination among the processes. Peer to peer computing allows users to interact with each other and find and share resources without requiring a centralized server. This gives rise to the need of a secure group layer which integrates authentication, admission control, authorization, fine-grained access control and key management. Currently there are some integrated solutions for secure group communication, but very few allow for the development of trust in a dynamic fashion. We propose a generic integrated framework for secure group communication in decentralized peer to peer groups with a hybrid access control model for peers in groups requiring multi-level access control. Our authentication scheme is based on self-signed certificates. Group composition and functionality of members in a group is dynamic and the group can dynamically prioritise requests for join. Our approach is adaptive, flexible and scalable and integrates aspects of incremental trust, along with role-based and policy based access control. The access policy framework allows peers in a group to dynamically change their levels, as well as modify existing rules in the policy.

1 Motivation and Introduction

Web based collaborative groups are becoming increasingly popular as peers with common interests form a network among themselves and automatically tend to create interest groups among each other, called communities. The group is governed by a set of rules that describe the conditions required to be part of the group.

Security in such dynamic collaborative groups is governed by membership control, authentication, access control and key management. Membership control and authentication is required to allow only authorized members to join the group. In the dynamic scenario where peers are constantly leaving and joining this becomes a difficult task. Another major challenge is access control since decisions need to be made based on collaborations from all peers and feedback mechanisms based on trust.

In the decentralized environment, peers in self-organising groups should have the right to collaboratively modify the access control policies governing their levels and the levels below them. Current solutions do not allow dynamic change of access levels or policies, with incremental building of trust during the communication. Nor do they allow dynamic authentication of participants who were previously part of the group and later wish to rejoin.

Motivating Application. The evolution of dynamic, self organizing groups like wikipedia, f/oss and other business community groups motivates new security requirements for peer to peer communication.

Consider an example of a developer’s community for some critical security related open-source software. The communication forum could be either chat or e-mail. New users would be allowed into this forum to put in their suggestions, only if they are introduced by an existing member. Depending on the sensitivity level of the code being developed, there could be hierarchical levels for the members of this forum. New members and existing
members could be periodically rated or evaluated by existing members based upon the quality of their contribution. A member receiving a high rating value could be elevated to a higher level in the forum. A peer or a member in any level need not always be evaluating other peers, i.e a member can have different roles in a group. For critical software like security applications, authentication of developing members would be important as well as deciding which member is allowed to play which role and participate in which level.

**Our work.** In view of the varied security requirements for diverse peer to peer applications, there is a need for an integrated framework for secure communication in dynamic groups with no dependance on a centralized server. We propose such an integrated framework, which has an adaptive, flexible, dynamic and totally decentralized access control mechanism, integrated with authentication and dynamic rekeying. The key features introduced in our framework are: There can be different levels within a group and peers at each level can have different functionalities. Peers in one level can dynamically elevate their levels in a group. Group composition determines join priorities. The authentication framework does not rely on X.509 certificates and is based on self signed certificates.

Section 2 gives the related work. In section 3 we give the architecture of our proposed framework. The detailed working of our protocols is explained in section 4. In section 5 we give the details of a prototype implementation.

## 2 Related Work

Security in collaborative groups is addressed by group membership, authentication, access control, and group key management. Role based access control models [3] [4] have been popular in group communication systems. Distributed establishments typically involve peers that do not know each other and have never met before. This brings in the concept of risk when peers perform transactions without knowing the reputation of those whom they are interacting with. This has precipitated work on trust and reputation mechanisms in peer-to-peer networks [5] [16] [7] [6]. Trust is a measure of how much a peer is willing to transact with another peer. It can be defined as a peer’s belief in attributes such as reliability, honesty and competence of the trusted peer[], either based on the peer’s own experiences or based on recommendations by other peers. Reputation [8] is one specific way of establishing trust. It defines an expectation about a peer’s behaviour, based on recommendations received from other peer’s or information about the peer’s past behaviour within a specific context at a given time.

Several efforts have been made towards securing group communication systems, for e.g Secure Spread [12], Antigone project [11] and Secure group layer SGL [13]. Secure Spread [12] is a secure group communication system that uses a fully distributed group key generation protocol, but it does not provide any authentication or group access control mechanisms and focusses primarily on LAN and interconnected LAN environments.

Antigone project [11] includes a flexible framework for secure group communication and utilizes a centralized member admission approach which is controlled by a session leader SL who interacts with an on-line trusted third party (TTP) in order to admit a new member. The TTP shares a symmetric key both with the SL and every potential new member. However Antigone is not designed for peer-to-peer networks.

Secure group layer SGL [13] is a secure group communication system aimed at WAN environments. It bundles a reliable group communication system, a group authorization and access control mechanism and a group key agreement protocol to provide a comprehensive and practical secure group communication platform. However the access control mechanism is not dynamic or scalable.
Some frameworks are focused on peer to peer applications. Kim et al [1] [2] proposed an admission control framework which revolves around two basic elements, viz: a group charter which has well defined admission policies and a group authority which is an entity that can certify group admission. However their scheme lacks the attributes of peers and cannot simplify authorization in collaborative environments. Further access control has not been considered, all members have equal access rights, and neither does it integrate admission control with group key agreement.

3 Proposed Integrated Framework

In the decentralized and dynamic peer group scenario it is necessary to have an adaptive access control mechanism where peers have the right to collaboratively make access control decisions, modify these decisions and frame rules based on attributes as well as trust evolved in the group. We propose a hybrid access control framework which integrates policy based access control and trust based access control along with multi-level access control.

3.1 Model Overview

We consider a peer based model where every peer $P_i$ has a unique user identity $UUID_i$ associated with it. A group is a named set of peers or a subset of a community that is governed under a set of rules that describe the conditions required to be part of the group, and is formed based on a particular interest criterion. The dynamic membership of the group implies that no peer will be highly available. The challenges are rapid admission of new members with appropriate access rights, dynamic elevation of access levels of existing members and dynamic revocation of access rights of leaving or defaulting members.

Basic Assumptions The following are the assumptions made in our model.

– No centralized control: Traditional access control models such as ACL or RBAC [3] rely on central servers for authorization. In our model a peer has a high level of autonomy and can frame and manage his own policies.
– Peers in the P2P system are loosely coupled and interacting partners are unknown.
– Peers in a group can belong to different levels within the group and can perform several roles.
– A role of a peer is independent of the level which the peer is in.
– Peers are individually capable of performing tasks of authentication, voting, access control, key management etc.
– Peers can compute their own public-private key pair and provide self signed certificates, binding their identity with their public key.

Each group has a group policy which describes the roles permitted in the group and the rules associated with each role. The functions that a peer is capable of performing in a group are storing and verifying certificates, authentication, voting, updation of levels, key management etc. The extent to which a peer performs these tasks is decided by the role the peer wants to play in the group. We identify the following roles:

1. **Member peer** A peer in this role participates in the activities of the group like gaming, conferencing, file transfer etc, but does not contribute to the admission of new peers, nor updation of levels of existing peers. Thus this peer need not store certificates of every other peer and need not invoke voting, either.
2. **Admission Control Peer** A peer with this role would participate in the voting algorithm, and would therefore store public keys of every other peer. However it need not store the updated ratings.

3. **Maximal role peer** A peer in this role would have all the functional components and would participate in updation levels of existing peers also.

The access policy would decide the role a peer assumes, based on the request of the peer and his credentials as well as rating calculated by the other peers.

### 3.2 Proposed Architecture

We present here a flexible architecture framework for secure group communication in decentralized peer groups, based on our proposed access control framework. The architecture is suitable for applications that involve sharing of high value resources, e-commerce applications, as well as general collaborations where security is not a major issue. The access policy framework is flexible and allows peers to change their levels as well as change the rules for membership within an existing level. Peers are rated based on their performance in a group along with contributory recommendations by other peers. A peer can then invoke the rekeying algorithm to elevate his level. Policy rules for elevating members from one level to another as well as for introducing new levels are dynamic and peers are able to frame and manage such policies.

![Integrated Framework](image)

A peer in a maximal role would be expected to have the following functional components:

**Attribute Manager**: Responsible for storing the attributes and certificates of the peers. The Attribute Manager is also responsible for updating the certificates of the peers with the new rating.

**Authentication Manager**: Verifies the validity of the identity and invokes the voting algorithm.

**Trust Engine**: Responsible for calculating the trust values of each peer, based on responses from other peers. It thus has a calculation engine and a feedback engine associated with it.
**Access Policy Manager**: Frames rules or policies based on attributes, behavior of peers and inputs from trust engine. It also checks the rules to allow members to join. It is called upon either by the Authentication Manager in case of a new member join, or the Updation Manager in case of an existing member level updation.

**Update Manager**: Responsible for checking the rating levels of peers after they have collected feedback from other peers and gives the input to the Access policy manager. It is also responsible for granting or revoking access rights to peers based on the dynamic policy.

**Key Management System**: Invoked whenever a new peer joins or an existing peer’s access level is changed depending on the dynamic policies, and is responsible for computing the group key. These components interact as shown in the figure 1.

### 3.3 Protocol Overview

Our initial membership admission protocol is similar to the protocol proposed by Kim et al. [1] and allows all peers to collaboratively decide whether a new user should be allowed entry into the group and if so decide the hierarchy level or role in which he can join the group. First time authentication is based on user identity, either an X.509 credential, or a self generated pseudo certificate and other attributes presented during initial admission request phase. This is handled by the Authentication Manager and Attribute Manager. If permitted to join, a peer is given a signed membership certificate which serves as an authentication token. For all practical purposes a new peer will be allowed to join a group at the lowest level initially. The membership certificate contains a field which has the trust level or rating of the member.

Subsequent per session authentication could be based on an authentication token received in prior session and recognized by all peers. This would require inputs from the Access Policy Manager and the Trust engine.

Once a member of the group, the access control policy could be dynamically changed and the user could be switched from a low hierarchy level to a higher hierarchy level depending on his performance in the group as well as the rating given to him by other peers in the group. A rating of a peer can either be periodically updated by all existing peers in the group or a peer can request for rating certificates from other peers in the group with whom he has had transactions.

Rekeying algorithm would be invoked by the peer whose rating increases, so that he could switch to a higher hierarchy. If a member who was previously part of the group and has left, wishes to rejoin the group, then based on the rating token and credentials that he possesses, he could easily rejoin without having to be subjected to the voting process, or if he has a high rating i.e a high trust level, then he may be permitted to join as a trusted member even though the votes of remaining members are less than the threshold i.e 50 percent i.e the admission policy changes if the member is an old member.

The credentials provided by the user could have various levels of trust, that could be incremental.

### 3.4 Policy Language framework

One major contribution of our work is the dynamic policy language framework which allows peers in one level to dynamically update to a higher level. A peer could also be collaboratively ejected from a higher level and forced to a lower level if his behavior in the group degrades. The group policy can also dynamically prioritise requests for join. Join priorities would depend on current group composition. Thus low level requests for join
could be postponed in a group which already has a large number of members performing the lower roles. It is also possible for peers at the highest level to introduce a new level into the group if the situation so demands. An example of this is detailed in section 5. To the best of our knowledge a group with such enhanced peer capabilities has not been discussed so far in the group communication systems.

We define the following components in our language framework:

- **Role Sets (RS):** This comprises of the different roles permitted.
- **Group Hierarchy (GH):** These are the different levels possible in the group and would depend upon the attributes of the peers and the composition of the group.
- **Service Set (SRVC):** This is the possible set of operations permitted within the group.

A JOIN_REQ or an UPDATE_REQ is defined as a triple \(< role, level, srv >\) where \(role \in RS\), \(level \in GH\) and \(srv \in SRVC\). Based on the join or update request, the system determines the applicable access policy for the requested service. The access policy for a role service pair is a set of clauses consisting of multiple access conditions. We use attribute-based credential specification. There is no compulsory reliance on X.509 identity-based certificates to encode user authentication information. The authentication credential comprises of self signed user attributes which are used for role assignment. The credential specification in our framework facilitates a combination of rule-based role assignment and role-based authorization. Roles in our model are of two types: *Attribute-Roles* and *Group Roles*. Attribute Roles correspond to the role the peer is assigned in the group based on his attributes and rating, thus in case of an University it could be the role of a Professor or a student or a Director. Whereas group-role corresponds to the functionality of the peer in the group, which the peer could choose at the time of joining and could modify later while in the group.

### 4 Working of Proposed Model

In dynamic peer groups the different cases needed to be supported are:

1. A New peer wishing to join an existing group
   - A peer having a X.509 certificate signed by a trusted CA.
   - A peer without certification from CA but who provides a self signed pseudo certificate with proof of possession of private key.
2. An existing peer wishing to have his level updated
3. A peer who was previously part of the group wishing to rejoin in a new session
4. Dynamic leaving of peers from the group.

We propose protocols to handle these different cases of join and leave based on our integrated framework. The **Notations** used in our protocols are:

- \(P_{new}\) : New Peer
- \(P_i\) : Existing Peer
- \(SK_i\) : Private key of i
- \(PK_i\) : Public key of i
- \(Cert_i\) : Certificate of i which contains the ID, Public key, Level/rating, Validity period.
- \{\text{msg}\}_{SK} : Signed message
4.1 Protocol for New Member Join

The creator or initiator of the group should publish and advertise his group along with the roles permitted and the group policy that specifies the parameters of the group, such as group type, admission policies, group name, group members etc. The group advertisement should be periodically broadcast by the peers of the group. The steps in the protocol for a new member wishing to join an existing group are shown in the figure 2. The detailed steps are explained below.

![Fig. 2. New Member Join](image)

1. **Group Discovery** A new peer would first contact the group service manager to obtain information about the existing groups. Once he finds a group that matches his requirements, he sends an admission request to a peer of this group.

2. **Admission Request** The admission request is a signed request which would include his credentials that he obtains from a CA say X.509 certificates or self-generated certificates, along with a request for the role that he wishes to join in, and the level that he wishes to join. Given that each peer has his own certificate which could be self signed or signed by a CA, a peer credential is created by hashing the concatenation of unique user ID UUID and public key fields and then signing this hash with the private key of the user $\{H[UUID||PK_{new}]\}SK_{new}$ and using this digital signature as the identity of the peer. This identity could be used as the peer’s credential in the messages. The certificate sent by the peer in this message contains his unique user ID, his public key, his initial rating and the digital signature created as his credential.

$$P_{new} \rightarrow P_i : \{JoinREQ\}SK_{new}, Cert_{new}$$

where

$$Cert_{new} = UUID_{new}, PK_{new}, Rating_{new}, \{H[UUID||PK_{new}]\}SK_{new}$$

For a new user his $Rating_{new}$ field will contain a single value signed by himself.
3. **Authentication** The peer receiving this signed request will obtain the identity of the requesting peer from the Certificate. If it is a signed certificate then the verification is easy. If however the Public Key pair is pseudo generated then the receiving peer will compute the hash of UUID and $PK_i$ and tally this with received message. If the two hashes match then the user is authenticated and will be granted access permissions based on the access control policy. If the user’s credentials contain a rating field with a single entry he will be treated as a fresh user and the voting process will be invoked by broadcasting this join request to the other peers. The certificate of the new peer is co-signed by the broadcasting peer, so that further verification of ID is not required by other peers. Thus our model achieves authentication by binding the unique identity of the user with his public key.

The authentication manager checks to see if old rating certificates exist, signed by previous group members. If not then the request is passed on to the authorization and access policy manager. If old certificates exist, and are authentic then the request is passed on to the access policy manager for new-session authentication.

$$P_i \rightarrow \text{All} P_n : \{\text{JoinREQ}\}_{SK_{new}}, \{\text{Cert}_{new}\}_{SK_i}, \text{VoteReq}$$

4. **Voting and Authorization** The peers all verify the authenticity of the new peer from his credentials. The peers return the results of voting to Access Policy Manager of the peer who had initiated it.

$$P_n \rightarrow P_i : \{\text{Vote, Level}\}_{SK_i}$$

5. **Access control** The Access Manager verifies the votes and if the votes are $> 50\%$ then the later gives a signed membership token to the new peer and calls the rekeying algorithm. Since this is signed with the group key of the current level $G_{key}$, which is unknown to the peer the later cannot modify it and forwards this directly to the Key Management System (KMS) for rekeying needed to join the group.

$$P_i \rightarrow P_{new} : \{GC_{new}\}_{G_{key}}$$

The entry of the new peer is broadcast to all peers of the group. The membership token $GC_{new}$ would contain details of the role granted to a joining peer, for e.g: member peer, admission peer or maximal role peer.

6. **Key Management** The Key Management component of the existing peer who receives the signed Group Membership certificate performs the group rekeying at the necessary level. TGDH [14] protocol can be used for computing the key.

Thus a new member wishing to join the group is permitted to join provided he is authenticated by an existing member, who is trusted by a certain threshold of members, say $50\%$ of the remaining members. The appropriate rekeying ensures that the new member and the existing members at that level share a key so that they can communicate securely.

4.2 **Protocol for Existing Member Level Updation**

Each peer has a local Trust calculator which is able to compute the rating of every other peer. A rating certificate can be used as a means of recommendation. This certificate contains the original trust value of the peer along with the recommended rating value given by each peer. The rating certificate would also contain an expiry date to prevent
the recommended peer from recycling good rating certificates beyond the specified period. When a certificate is about to expire, the peer can contact the recommending peer and get the validity of the certificate extended. Thus for each peer it has interacted with, the peer needs to keep a copy of the certificates it issued to that peer for validation purpose and for trust and contribution score updating. In addition a peer could maintain a revocation list of peers who are not to be trusted and periodically this information could be exchanged amongst peers. The fields in the rating certificate are Recommending peer’s identity, Recommended peer’s identity, Original trust value, Issuing date and time, Expiry date and time, Contribution score and Signature of recommending peer. The steps in the protocol as shown in the figure 3 are detailed below.

1. **Request for Rating** Once a member of the group, a peer can periodically request the members of the group to rate him based on his performance and the trust level he has already acquired. Appropriate access privileges are granted to a peer based on a scoring system.

2. **Rating Feedback** The recommendations collected by a peer from other peers are all signed by the recommending peer thus preventing a peer from being able to modify the recommendations/ratings given to him by other peers.

3. **Request for Updation** The peer wishing to get his level updated would submit an update request which consists of the role he wishes to play, the level he wishes to join in, along with his accumulated ratings stored in his certificate.

$$P_i \rightarrow P_j : \{\text{UpdateREQ}\}_{SK_i}, Cert_i$$

where $Cert_i$ contains rating certificates from other peers. $P_j$ computes the global rating value after verifying the authenticity of individual ratings. The Updation Manager interacts with its local Trust Engine to calculate the average rating of the peer. It then forwards this request with rating score to the Policy Manager.

4. **Access Control** Access rights will be assigned based on a combination of the credentials, self rating and behavioral rating as well as rating given by other peers. Trust
threshold, denoted as \( A_{ij} \) is a summation quantity. Any peer who has its overall trust value and overall contribution score equal to or greater than the corresponding thresholds, can get the corresponding hierarchical access right. The policy would be flexible so that it could be modified contributively above a certain level. Thus maybe peers at access level 3 could modify the policy rules for entry into that level. The details of the algorithm used to calculate the rating is given below.

**Rating Algorithm** For calculating the trust/reputation rating we use the concept proposed by Eigen Trust \([9]\) with some modifications. We state that the factors on which reputation depends are Peer Feedback, Weighted Cost of transaction and Credibility of peer who is giving the rating.

Thus each peer has a local trust value, based on his initial credentials and his performance in the system, as well a global trust value which is based on the ratings provided to him by other peers.

**Weighted Cost** We introduce the concept of weighted cost to overcome the drawback of the eBay \([15]\) reputation system where buyers and sellers can easily default on large orders after gaining a good reputation value by successfully completing small orders. We put an upper bound on transaction as \( \Theta \) and state that for all transactions whose cost or weight is \( < \Theta \) the reputation is calculated normally but for every transaction with weight \( > \Theta \), the reputation value is multiplied by a constant which is a multiple of this threshold value. This would ensure that smaller transactions generate lower trust value than larger transactions.

Thus local rating or reputation index \( RI \) for a peer \( i \) is calculated by peer \( j \) as follows:

\[
RI_i = RI_j \times \sum T_{ij} \times Trans\_cost
\]

for transactions whose cost or weight is \( > \Theta \). where \( T_{ij} \) is the opinion peer \( j \) makes about peer \( i \) per based on transactions and behavior.

The global rating of Peer \( i \) is then a summation of the local ratings computed by all the peers it interacts with. A peer cannot modify its trust and contribution rating even though it stores the rating locally in the form of certificates.

5. **Key management** The necessary rekeying is done by the Key Management system. In order to ensure multiple levels of access privilege for group members, there can be separate key trees for users with same separate access privileges i.e each hierarchy of users would have a separate group key for encrypting their data. This can be done by calling the key agreement protocol separately at each hierarchial level. Thus each user would have to maintain different sets of keys and when a user switches from one level to another, he could call the rekeying algorithm to ensure that backward secrecy is maintained. We propose a key management graph similar to the one in \([10]\). However the rekeying algorithm is not a one way function controlled by a KDC, but is a variant of the TGDH algorithm.

The signature of the recommending peer on the rating certificates, provides the necessary authentication while updating peers, and also prevents a peer from modifying his own rating. Necessary rekeying is done during updation thus ensuring that backward secrecy is maintained.

4.3 **Per-Session Authentication of Previous Member**

Authentication of a peer is based on his credentials, authentication token received in earlier sessions, and/or rating granted to him in earlier session. The role assigned to him is based
on his attributes and the rating. It would be the task of the access policy manager to check whether the old rating certificates have expired or not or whether, the members who had given the rating earlier are revoked or not.

4.4 Dynamic Leaving of Members

When a peer leaves a group, it may not be identified immediately. Peers may send a query message to find out when a particular peer last communicated. If the peer is still a current member of the group, response would be immediate. However, if a certain threshold of members reply stating that the last communication was not very recent then the group may assume that this particular peer is no more a current member of the group. Periodic rekeying or batch rekeying could be another solution to tackle security threats from peers who leave without informing.

5 Implementation of Secure Multi-Chat

We implemented a prototype chat application based on peer to peer framework, where all the system components and processes described earlier have been implemented. For this sample application the components of our policy language were as follows:

\[
\begin{align*}
RS &= \{ \text{Director, Professor, T.A, Student} \} \\
GH &= \{ \text{level 1, level 2, level 3} \} \\
SRVC &= \{ \text{message_send, message_receive} \}
\end{align*}
\]

The group roles are member, admission control peer and maximal role peer. Some sample rules for this application are illustrated below:

**Rule1:**
- If request = JOIN_REQ then
- if role = member \(\land\) level = 1 \(\land\) srv = mesg_receive then
- if rating > 1 \(\land\) voting minbound > 50 then permission = “grant”

**Rule2:**
- If request = JOIN_REQ then
- if role = member \(\land\) level = 2 \(\land\) srv = mesg_receive, msg_send then
- if rating < 4 then permission = “deny”

**Rule3:**
- If request = UPDATE_REQ then
- if role = admission_peer \(\land\) level = 2 \(\land\) srv = mesg_receive then
- if rating > 6 then permission = “grant”

**Rule4:**
- If request = UPDATE_REQ then
- if role = maximal_peer \(\land\) level = 3 \(\land\) srv = mesg_receive then
- if rating > 8 then permission = “grant”

**Technologies used**

Java programming language was used for implementation of all components. We used JXTA as the base platform for application development 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. JXTA uses a small number of protocols, is easy to implement and integrate into P2P services and applications. It is independent of transport protocols and can be implemented on top of TCP/IP. The main components in a JXTA network are: Peers and peer groups,
Messages, Pipes, Services and applications, Advertisements. JXTA divides software into 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.

Currently we have used XML to specify set of rules for policy language framework, as it provides a customizable and extensible framework to represent messages.

We implemented a secure chat application based on our framework. A new peer wishing to join the group has to specify his attributes and the role. His unique user identity is bound to his public key and is used for authentication. Thus a peer with the same UUID cannot pose as someone else and try to join the group. Multiple peers in the same level can send messages to each other and receive them while peers at a lower level are not able to read those messages. After joining the group the peer broadcasts his public key to other members of the group. Public keys of all peers are stored by the Maximal role peers and Admission control peers. A peer in a group can ask other peers to rate him and can ask for an updation of level when suitable rating is obtained. The dynamism in our policy can be explained with the help of an example. Consider a University running UG courses with Directors playing the role of maximal peers at level 3. If a PG course is now introduced in this University then maximal peers at level 3 can collaboratively decide to change the group policy and introduce a new level.

6 Conclusion and Future Work

Currently there are some integrated solutions for secure group communication, but very few allow the development of trust and switching of users between hierarchical levels in an incremental fashion. We have modeled an integrated framework for decentralized groups, which allows us to implement secure communication between different peers using admission and trust based multi-level access control along with key management. Our dynamic access control policy permits peers to collaboratively frame, and modify policies at various levels. The policy also permits peers to update their access rights within a group. Join priorities can also be decided by the group policy and depend upon current group composition. Idea of per-session authentication is also proposed which depends on the incremental trust factor and eliminates the need of the voting process for a old member of the group who wishes to re-join. We have successfully implemented a prototype chat application of our integrated framework based on JXTA.

Though we have not done a detailed security analysis of our protocols, we have taken care to see that the necessary authentication is done when a new member wishes to join (section 4.1) or an existing member requests for level updation (section 4.2). Secrecy in different levels is maintained by an efficient key management algorithm. Peers in different levels maintain different sets of keys and encrypt data with an appropriate group key. Backward secrecy is maintained by doing rekeying at appropriate places.

A more detailed security analysis would involve taking cognizance of the threats to this model. Common threats from malicious peers, free-riders and whitewashers have been discussed by J. Yoon and A. Ravichandran [6]. Many of these could easily be handled in our framework. However threats posed by collusion of existing peers needs detailed analysis and is in the scope of our future work.

We would like to focus on the policies for composition of groups in the dynamic scenario. Future work would also include measuring the behavior of the group, with varying
group compositions. Measuring intra-group trust level and a focus on the behavior of the group in presence of malicious peers are the other directions of work proposed.

References

7. Thomas Repantis, Vana Kalogeraki.: Decentralized Trust Management for AdHoc Peer-to-Peer Networks. MPAC, November 27-December 1, ACM 2006