Secure Group Communication

Btech Project, Phase 1

by

Ajit Burad
Roll No : 03005009
Computer Science and Engineering
Indian Institute of Technology
under the guidance of

Prof G. Siva Kumar

November 13, 2006
Abstract

Peer to Peer communication is emerging as technology having a high potential to provide scalable and flexible structure to communicate, sharing and in 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 as we need a group key so that all messages are encrypted with it thus groups secrecy and confidentiality is maintained. This report provides an idea of what types of policies we need to make secure communication which is also flexible. We describe how a policy language based framework can be designed which lead to dynamic and flexible architecture which support group communication using role and attribute based access control. We will implement an prototype application based on peer to peer framework (like JXTA) to demonstrate above set of policies.
Chapter 1

Introduction

The secure group communication abstraction provides both point-to-point communication and secure multipoint communication. This incremental and dynamic growth pattern is not well supported by a rigid server-based structure. Thus, the natural alternative to a server-based model is to provide a reliable and secure group communication infrastructure to support a dynamic and scalable peer-to-peer model. This project deals with designing a framework for peer to peer communication along with admission and access control with secure communication. Peer groups 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 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 like:

- Teleconferencing
- Chat rooms
- Collaborative groupware
- Multi-user Games
- Replicated Servers.
1.2 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 intensional DoS (Denial-of-Service) attacks. While Server-client architecture can be in trouble if servers are down due to load. Also as large number of clients are connecting to same server, we need high performing machines at server end, thus is more costlier as compared to P2P architecture.

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 each other but we will require that number of message exchanges should be minimum. Along all these feature, for a secure group communication we will require that all traffic is encrypted using some group key thus maintaining group security and privacy.

1.3 Overview

In this report we present how peer groups uses admission and access control polices to provide secure group communication. This report sketches steps involved in admission of new peer to group. We also discuss how key management works as for security and privacy purposes it is needed that all messages are encrypted which is done using secret group key. Then we provide an brief overview of JXTA, which we are going to use for our implementation part. Aim of this project is to provide a framework, based on policy language which define access rules for peers. We also look in role based and attribute based approaches for access control. In this we also provide an hierarchical arrangement based on role assigned to peers as each peer has some different set of attributes and properties.
Chapter 2
Admission and access Control

In Peer to Peer group scenario we will require that there are some standards which define which user can join the group and once they are granted admission we will again want 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 admission of new peer to group whereas access control policies play important role in secure access to group messages and resources. So here I present a description of what admission and access control is.

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 criterions are.) So group information and admission rules must be well electronically documented. Ways in which admission control can be achieved:

- **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 som 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 enter 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 this project we will study this scheme more deeply where there is contribution by all and finally we will implement such policy based on JXTA (P2P framework.

2.2 Admission Process

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

• 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 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 respons 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.

• Request for joining group :
   Now there can be two mechanism for the new member to spread the word that he want to join the group.
   
   – One choice can be that he sends message to all other peers in the group.
   – Or the framework is so that new member will request one of the peers \( P_1 \) for his joining. Now \( P_1 \) can communicate this to all other members of the group.

• Authentication of user who wishes to join
   The new peer must present some credential ( like in IIT we can use LDAP for this purpose) which describes one or more properties / attributes of the owner asserted by the user, signed with the private key of thone user.

• Collaborative decision by majority peers as to whether to allow new peer to join or not.
   
   – Previous members may decide based on majority of votes whether to allow new member.
- Static ACLs (access control lists)
- Admission by some of authoritative servers.

- Issuance of some token which could serve as a qualifier.
Once enough votes are collected, Group Authority verifies all the votes, and decide whether to accept the new node as a member. If the requester is qualified, A new group key is generated and the the authority peer will issue the new key to it and update the related peer group information. Having the new group key, the new node can join the secure peer group.

- Assigning role to members according to credential and login information.

### 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. Now we can divide group based on roles. It can also help us to have weighted decision in some cases. (fig2.1)

- **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 sometimes identity-based schemes do not provide enough flexibility. If the 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. And as the number of attributes involved increases, number of roles and permission needed to encode these attributes increases exponentially. Also RBAC might need some centralized server to manage user to role and role to permission assignment.

Figure 2.1: Role Based Access Control
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 a authentic member will know the secret symmetric key thus only he can decrypt the message.

Security requirements in key management:

- **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 rekeying. To summarize rekeying 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 rekeying operation is needed each time a new group member joins.

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 Centralized versus distributed key management

- **Centralized** : A central authority distributes group keys to group members. This central authority can be Trusted third party or can be group authority.
• 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 members cooperate to establish a group key. Here we will discuss some of the group key management protocols used in distributed systems.

• Ring Based Approach (using Group Diffie Hellman): In this protocol, members are organized into a virtual ring, so member $M_i$ communicates 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) Here members of a group are organized at leaves of the tree. Each leaf holds its secret key, and it calculates $g_{ki}$ and propagates towards its parent. Now a combined key from children’s $g_{ki}$ and $g_{ki+1}$ is calculated so that the end group key is calculated at the root. In case of membership change (join/leave) the tree is rebuilt consequently and hence all the members update the group key which is the new key associated with the root of the tree.

![Figure 3.1: Skinny tree protocol](image)

• Broadcast based approach: This approach relies on broadcasting secret messages and calculating group secret key which is a 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 calculates $S_i = g^{(r_i)} \mod p$, where $g$, $p$ are pre-shared prime numbers. So each sends its calculated $S_i$ to every other peer. Now each peer can calculates group key, $K = g^{r_1 \cdot r_2 \cdot ... \cdot r_n}$. 
Chapter 4

Peer to Peer framework

This chapter provides a brief overview of JXTA framework. In this project we plan to use JXTA as base platform for application development which allows us to incorporate admission and access control along with secure key management for group communication. So here is brief introduction of JXTA.

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.

Here we aim to use JXTA as programming platform and create a test application which uses policy language (which we have to define so that we can define policy rules for a given scenario) for admission control. At an initial level we will try to implement password authentication or admission by voting of group members.

Where can XML be well used:

- Messaging customizable, extensible, RPC
- Data Storage searchable, validation
- Resource Description peer, peer group, service,
- Distributed Search JXTA Search

All JXTA entities are described using advertisements with expiry. Advertisements and messages are in XML. In our project scenario XML can also be used to represent policy language.
Chapter 5

Proposal

Consider an multi-chat application in IIT implemented using group communication. Multiple session exists, each implemented as a group. Each session could have different class of members. So in such a scenario various users may join a session in different roles e.g professor, student, T.A., Director, Head of Department etc. So in a group users belonging to different categories must be assigned different roles. So we need to define some set of rules which decide who can start the group, how to decide upon whether to allow new user. We want to develop a hierarchical schema in group.

We will also ant that all message exchanges are done securely, therefore we want our messages to be encrypted. So we will attempt to implement a hierarchical key management protocol, as we may want different roles should have different set of permissions.

Here using JXTA as base platform we propose to build a framework which allows group communication along which set of policies which controls admission an access control. Also in case peer groups we will want our policies to be dynamic. We will attempt to develop a policy language which standardize how rules are defined. We may want out language to be English to define rules but that may be difficult to implement and to be understood by other peers, also difficult to agree on same format to write some specific rule.

Example of some rules can be:

1. allow(enter site()) ← declaration(user = U, passwd = P), has passwd(U, P)
2. allow(access resource-id ) ← user_rating ≥ 6
3. allow(browse(articles)) ← user(X), credential(C), states(C, member(X, group_id))

When an user makes a request to join group it provides some credential and attributes which can be authorized. An appropriate role may be assigned to peer. Policies may map access permission to different roles, thus here access rules are mapped to roles (different hierarchies), not to individual peers.
5.1 Conclusion

5.2 Future Scope
Bibliography


