Admission and Access Control in Secure Group Communication

Mini Project Report

by

Pratyush Chandra
Roll No: 03D05003

under the guidance of

Professor Bernard Menezes

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
November 14, 2006
Acknowledgement

I would like to thank my guide Prof. Bernard Menezes and Madhumita Chatterjee for their guidance and support in making this mini project an informative experience for me.

Pratyush Chandra
Abstract

Secure group applications and services are very common in today’s computing. The popularity of the secure Peer-to-Peer (P2P) group application prompts the need for specialized security services which makes P2P group security an important and challenging research topic. To ensure secure communication between peers in the group, messages are encrypted using a secret key that is shared only by the group members and thus key management becomes an integral part of secure group communication (SGC). However, an important pre-requisite for many P2P security services is secure admission, or how one becomes a peer in a P2P setting. Also many group communications require a security infrastructure that ensures multiple levels of access privilege for group members. Access control in hierarchy is prevalent in chat applications, which consist of users that belong to different categories. In this report, we propose a admission and hierarchical access control based framework for secure group communication. We also introduce our implementation experience.
## Contents

1 Introduction 2

2 Group Key Management Protocols 3
   2.1 Description ................................................... 3
   2.2 STR Protocol (Distributed Key Management Protocol) .................. 4
      2.2.1 Group Initialization ........................................ 4
      2.2.2 Member Join ............................................... 5
      2.2.3 Member Leave .............................................. 6

3 SGC based on hierarchical access control scheme 7
   3.1 Background .................................................... 7
   3.2 Proposed Scheme ............................................... 7
      3.2.1 Identity Verification ...................................... 8
      3.2.2 Joining and Admission Control ........................... 8
      3.2.3 Access Control ............................................. 9
   3.3 Implementation ................................................ 9
      3.3.1 Group Initialization ...................................... 9
      3.3.2 Searching Group Advertisements .......................... 9
      3.3.3 Processing Join Request (Phase I) ....................... 9
      3.3.4 Processing Join Request (Phase II) ..................... 10
   3.4 Few Implementation Issues .................................... 10
   3.5 Conclusion ..................................................... 10

4 Appendix 12
   4.1 JXTA Overview ................................................. 12
Chapter 1

Introduction

With the progress in the technologies underlying multicast networking has led to the development of many group-oriented applications, such as video conferencing, multi client chat, and gaming. But with increasing popularity of such group-oriented applications, security and efficiency in communication has become an important issue.

First step towards creating a secure group is the admission of a peer. In other words how one becomes a member of an existing peer group. Without a secure admission process based on true identity of joining member there is no point in using secure key management and trust management since a malicious prospective member can easily generate any number of false identities. For this reason, admission control [5] is the first crucial step.

Now once a member is authenticated to join a peer group, peers need to communicate among themselves in a secure manner. This is achieved by encrypting the communication packets which is shared by members of such a peer group. Thus efficient and scalable key management protocols become an integral part.

While joining the group, peers may subscribe to join at different hierarchical levels. For ex. in a multi chat application, a user may join as student, faculty or as director based on their identity. Thus, it is necessary to develop group access control mechanism [2] that supports the multi-level access privilege, which is referred to as the hierarchical access control.

This paper is organized as follow: Chapter 2 gives an overview of Group Key Management protocols. Chapter 3 discusses about our proposed architecture.
Chapter 2

Group Key Management Protocols

2.1 Description

Protocols used for maintaining and distributing group keys are termed as Group Key Management Protocols [4]. Group keys can be used for making the communication secure among group members. But the main issue is distributing the cryptographic keys. For this various protocols have been proposed. They can be broadly classified into three categories:

- **Centralized group key management protocols**: A single entity is employed for controlling the whole group, hence a group key management protocol seeks to minimize storage requirements, computational power on both client and server sides, and bandwidth utilization.

- **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 key management protocols**: There is no explicit Key Distribution Center (KDC), and the members themselves do the key generation. All members can perform access control and the generation of the key can be either contributory, meaning that all members contribute some information to generate the group key, or done by one of the members.

There are some requirements which should be taken into consideration by any group key management protocols. These requirements are enlisted below:

- **Forward Secrecy**
  Leaving member should not be able to access any future key and hence unable to decrypt any communications. This ensures that a member cannot decrypt data after it leaves the group. To assure forward secrecy, a re-key of the group with a new group key after each leave from the group is the ultimate solution.

- **Backward Secrecy**
  Any new joining member should not have access to any previous keys. This ensures that
a member cannot decrypt data sent before it joins the group. To assure backward secrecy, a re-key of the group with a new group key after each join to the group is the ultimate solution.

- **Collusion 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. Service like multimedia applications are sensitive to jitters and delays in packet delivery. Therefore, any key management scheme should take this into consideration and hence minimizes the impact of key management on the delays of packet delivery.

In next section we describe the STR protocol [1] which falls in distributed key management protocol category. In our proposed framework we have used STR protocol for providing security and hierarchical access control. Main reason for incorporating this protocol is that it takes very few steps and much less time in rekeying operation.

2.2 STR Protocol (Distributed Key Management Protocol)

STR is distributed/contributory protocol which is primarily variations of the n-party Diffie-Hellman [3] key agreement/exchange. In this protocol every member maintains an unbalanced tree view for all the current members. For n members height of the tree is n-1 (See Figure 2.1). The key tree is organized as follows: each node is associated with a key and a corresponding blinded key. The root is associated with the group and each leaf with a distinct member. The root key represents the group key shared by all members, and a leaf key represents the random contribution of each member. Every member knows the structure of the tree and its initial position within the tree. Further each member knows its session random and the blinded session random numbers of all other members. The protocol relies on the fact that every member can compute group key if it knows all blinded keys in the key tree.

2.2.1 Group Initialization

Let $M_i$ represent represent $i^{th}$ member. Each member computes secret random $r_i$ and its public blinded version

$$br_i = g^{r_i} \mod p \quad (2.1)$$

where $g$ and $p$ are DH parameters. Each internal node holds a secret random $k_i$ and its blinded public version

$$bk_i = g^{k_i} \mod p \quad (2.2)$$
The secret key is result of a Diffie-Hellman key agreement between the node's two children. Each secret $k_i$ is recursively calculated as follows:

$$k_i = (bk_{i-1})^r_i \mod p = (br_i)^k_{i-1} \mod p = g^{r_i * k_{i-1}} \mod p \text{ if } i > 1.$$  \hspace{1cm} (2.3)

So initially $M_1$ and $M_2$ compute group key corresponding to $k_2$. $M_1$ computes as follows:

$$k_2 = (br_2)^r \mod p = g^{r_1 * r_2} \mod p, \ bk_2 = g^{k_2} \mod p$$
$$k_3 = (br_3)^{k_2} \mod p$$
$$k_n = (br_n)^{k_{n-1}} \mod p$$  \hspace{1cm} (2.4)

Thus final group key computed is

$$g^{r_1 g_{r_2 - 1} \cdots g_{r_1}^2}$$  \hspace{1cm} (2.5)

Then $M_1$ broadcasts all $bk_i$’s to all members and each member then calculates $k_n$. Each member should store and maintain all the public keys associated to all the nodes of the tree.

### 2.2.2 Member Join

Suppose now $M_5$ wishes to join. Following steps are performed:

1. $M_5$ broadcasts $br_5$ to all group members.

2. Each member then computes new key.
3. $M_4$ which is the right most leaf node takes in charge and sends all previous $br_i$’s and $bk_i$’s to $M_5$ who then computes new key $k_5$.

### 2.2.3 Member Leave

1. If $M_n$ leaves then $M_{n-1}$ takes responsibility to create new key.

2. Everybody renumbers the node.

3. $M_{n-1}$ selects new random key $r_{n-1}$’ and computes all $bk_i$’s and broadcasts it to all members.

Thus, incase of membership change (join/leave) the tree is rebuilt. Due to linear structure of tree, this protocol induces $O(n)$ key calculation. Also notice that join requires two communication rounds while leave requires one round.
Chapter 3

SGC based on hierarchical access control scheme

3.1 Background

Consider a scenario of multi chat application used in a university. A secure group is formed for this application. User may join this group based on their identity say Student, Professor and Director.

Admission control requirements are as follows:

- Identity of joining member should be verified. For ex: Student may impersonate as a Professor to gain more privileges.
- Majority of the current group members should agree on new members join request.
- A high hierarchy member may wish to join group at low hierarchical level. This should be acceptable. For ex: A Professor may wish to join in student.s hierarchy and converse with them.

Access control requirements are as follows:

- If a high profile user wants to join the group then he should be directly allowed. These rules are defined in static admission control policies. For ex: Suppose currently only students are engaged in conversation and some Professor wants to join the group then he should be allowed straight away.
- Conversations between one hierarchy members should not be understandable by other hierarchy. For ex: If few professors are conversing then students should not able to understand them.

3.2 Proposed Scheme

In this section we propose a scheme which meets all the admission and access control requirements mentioned in previous section. Figure 3.1 explains the overall steps.
3.2.1 Identity Verification

We assume that there is a central server which maintains a list of all the Students, Professors and Directors. This server also keeps the public key of all the university people. People inside university can contact this server and verify anybody's identity as well as get public keys.

3.2.2 Joining and Admission Control

Suppose a new peer (say N) wishes to join a group and communicates with an existing member (say M) of that group. M firstly verifies the identity of N. On successful verification, M checks if the N’s hierarchy level is higher than any of the existing members. If it turns out to be true then voting is not considered. Based on static access control N is directly allowed. In other case, M initiates polling and collects votes from all the current group members. N is allowed to join only if total positive votes received are greater than a specified threshold T (say 50%) defined by group admission policy.

Now if N is allowed by group consensus or based on static access control policy then a certificate is sent to the requester by group administrator G (Member who first created the group and defined the admission control policies). Using this certificate as credential N is allowed to join the group.
3.2.3 Access Control

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 $K_i$.

We have used STR distributed key management protocol as the group key management protocol. When $N$ joins the group, controller corresponding to level $H_i$ manages the rekeying operation. Also when any member from level $H_i$ leaves the group rekeying is carried out by level $H_i$ controller. We choose STR as our group key management protocol as it totally decentralized and based on equal contributions from all the members. It provides basic requirements like forward secrecy, backward secrecy and key independence. It also requires smaller number of unicasts and multicasts to compute a new group key after a member leave or join. Hence it is quite efficient for such a dynamic group.

3.3 Implementation

We implemented these admission and access control policies using Java programming language. Our application is developed on top of JXTA technology [6, 7]. JXTA provides base level framework for forming virtual P2P groups, advertising and receiving events. For overview of JXTA refer appendix 4.1.

3.3.1 Group Initialization

Group administrator $G$ initializes a new secure group and regularly publishes the group advertisement into the network. The group advertisement contains description and membership policy of that group.

We defined a new membership policy based on X509 certificate verification. Since a X509 certificate is issued as credential by group administrator $G$, the admission control is controlled by this new membership policy. Also the X509 certificate sent by $G$ is encrypted with RSA public key of requester $N$. This RSA public key is retrieved from the central server.

3.3.2 Searching Group Advertisements

Each member in the secure group broadcasts the group advertisement as well as personal advertisements. Personal advertisement contains the contact information of that peer. So if a new member $N$ finds a personal advertisement from a existing member $M$ of group then $N$ can directly send a join request to $M$.

3.3.3 Processing Join Request (Phase I)

On receiving join request $M$ communicates with central server for identity verification. Based on static admission policy either $N$ is directly allowed or voting is carried out. Both the processes,
contacting central server and voting is carried out in a secure manner by encrypting communication with receivers RSA public key. If N is allowed then G issues a X509 certificate to N. N plugs this X509 certificate into the membership service of group. The membership service verifies whether this certificate is issued by G only. On successful verification N can listen and send messages within the new group joined.

3.3.4 Processing Join Request (Phase II)

Issuing X509 certificate is only one half the portion of member join. On a successful member join using X509 credential, group rekey at level $H_l$ is performed by a series of unicasts and multicasts. Finally member communicate using DES encryption with key $K_l$ corresponding to their level. This completes the join process.

3.4 Few Implementation Issues

Although unicasting is made very efficient in JXTA, multicasting is not very reliable. Many a time multicasts is not received by group members. This makes the rekeying algorithm as well as communication messages slow and error prone. Also due to restriction in defining membership policy, currently only group administrator G is allowed to issue X509 certificates to new member N. Otherwise, we would have made admission control totally decentralized.

3.5 Conclusion

This paper presents a decentralized framework for providing static admission and access control. We have attempted to make the framework as decentralized as possible, which avoids single point of failure and enforce equal trust among group members. Also instead of creating different subgroup for various hierarchies to access group services, we have tried to enforce the same by maintaining separate key at different levels. This saves the time against any new subgroup formation in real time application for new group services. In future, we would try to incorporate dynamic admission and access control policies into our existing framework. Also currently rekeying is done at particular hierarchy level $H_l$ of joining/leaving level. This can be modified to series of rekeying for all the levels $H_i \leq H_l$. This will enable all the low hierarchy level group services accessible by group members.
Bibliography


Chapter 4

Appendix

4.1  JXTA Overview

Introduced by Sun Microsystems, Inc., the award winning JXTA technology is a set of open, generalized peer-to-peer protocols that allows any connected device (cell phone to PDA, PC to server) on the network to communicate and collaborate. Project JXTA is an open source effort that involved the developer community from the start.

The project is first generalizing of P2P functionality and then building core technology that addresses today’s limitations on P2P computing. The focus is on creating basic mechanisms and leaving policy choices to application developers. JXTA technology leverages open standards like XML, Java technology, and key concepts that make the UNIX operating system so powerful and flexible including the ability for shells to connect commands together using pipes to accomplish complex tasks. By using existing, proven technology and concepts, the result will be a peer-to-peer system that is familiar to developers and easy to build upon.

Project JXTA is creating a peer-to-peer system by identifying a small set of basic functions necessary to support P2P applications and providing them as building blocks for higher-level functions (Figure 4.1). At the core, capabilities must exist to create and delete peer groups, to advertise them to potential members, to enable others to find them, and to join or leave them. At the next layer, the core capabilities can be used to create a set of peer services, including indexing, searching, and file sharing. Peer applications can be built using these facilities. In addition, peer commands and a peer shell have been created to create a window into the JXTA technology-based network.

![Figure 4.1: JXTA software is layered with core functionality developed through an open, collaborative effort, and higher-level services developed by P2P community developers.](image-url)

Figure 4.1: JXTA software is layered with core functionality developed through an open, collaborative effort, and higher-level services developed by P2P community developers.