Access Control in Group Communication Systems

Qi Li[1], Mingwei Xu[1], Xinwen Zhang[2]
[1] Department of Computer Science, Tsinghua University, Beijing, China
{liqi, xmw}@csnet1.cs.tsinghua.edu.cn
[2] Samsung Information Systems America, San Jose, CA, USA
xinwen.z@samsung.com

Abstract—**With advances in distributed computing technologies, Group Communication Systems (GCS) have received a lot of attentions. Besides reliable and ordered message delivery services, these applications require plenty of security services, such as data secrecy, data integrity, and user authentication. However, less work has been invested on how to integrate authorization scheme within efficient communication systems, especially for group collaborations. In this paper, we present a flexible and efficient authorization scheme which provides group-level fine-grained access control and can be easily integrated to existing GCS. More specifically, we propose the concept of Virtual Group (VG) and automatic access control policy generation mechanism to realize secure collaborations between different groups. We implement a prototype with Spread and our experimental results demonstrate the efficiency and scalability of our authorization scheme.**

I. Introduction

Group Communication Systems (GCS) have gained considerable attention because of their controllable distributed message delivery capability, which is essential to distributed applications that require reliable message delivery and high availability service such as enterprise video conferences and virtual communities. Since these applications are expected to run over the Internet, security in these applications is an important issue. To this end, research community have invested a lot of efforts in developing integrated security technologies into distributed systems, e.g., several works have been proposed to design scalable and fault-tolerant group key management protocols [4].

However, there has not been enough research into the authorization scheme to enforce fine-grained access control into GCS. Since GCS is used on general-purpose platforms with different security requirements, a flexible and scalable access control service is essential, especially for dynamic group collaborations. An access control framework for GCS is proposed in [12], but this work fails to provide real implementation architecture and mechanism. In addition, there are some existing work providing access control for grid computing [14], [1], [10]. Authorization decisions in these approaches are based on static subject attributes such as group (or organization) memberships, and can not make access decisions for dynamic group collaborations.

Role-based access control (RBAC) and its variants have been proposed and deployed in many systems, which have been proven to be able to simplify security administrations and provide efficient policy management [6], [8], [18]. The essential concept of RBAC is to define roles each of which is a collection of permissions that can be invoked to access protected resources. A user can be assigned to a set of roles such that obtain the permissions of the roles. One of the main motivations of using RBAC is that it can simplify security administration in large-scale organization environment. However, traditional RBAC model focuses on single and closed security domain, where a user's roles are pre-defined [17], [13] based on her unique identity. Thus they cannot be directly used in GCS environments.

In Internet-based GCS systems, individual groups have their own users and corresponding permissions. For a collaboration, users from various groups perform different tasks in the collaborative job. For security purposes, sensitive operations within a group have to be controlled, e.g., based on a user's duty in the group. The challenges of this problem lie on the features of group-based computing environments, such as dynamic user activities, temporal permissions for a user and temporary and ad-hoc collaborative tasks.

Towards these challenges, we propose a group-based RBAC (GB-RBAC) model to develop a scalable, flexible, and efficient authorization scheme in dynamic and collaborative GCS environments. Our model not only provides access control for distributed applications and decentralized management, but also introduces automatic policy generation to handle dynamic group collaborations through the concept of virtual group (VG). Specifically, a VG is created whenever a collaboration is needed between existing groups and its users are automatically exported from cooperative groups through an algorithm. Within it, a group administrator defines roles based on permissions, which are the interfaces to perform sensitive operations in the VG and assigns users to the roles locally. We have implemented a prototype system based on *Spread*, which provides application-level multicast, group communication, and peer-to-peer message services [4]. The performance evaluation results show that our proposed scheme are efficient and scalable.

The paper is organized as follows. Some related work are presented in the next section. Section III introduces GB-RBAC model and the concept of Virtual Group. Decentralized authorization schema in GCS are illustrated in Section IV. We describe our implemented system with proposed scheme in Section V. Section VI present the conclusion of this paper.

II. Related Work

There have been many research work on cryptographic key management and protocols for secure group communication [2], [9], [15], [19]. The key problem is the confidentiality of data with dynamic user leaving and joining in a group. However, there are much less efforts on access control and authorization management in GCS. Antigone [11] is a general security mechanism for GCS with which various access control policies can be implemented, such as session management and user membership. Amir *et al* [3] discuss the separation of authentication and access control in Spread. The access control module in Spread can enforce traditional identity-based, role-based, and credentials-based policies. However, it does not support dynamic access control policies such as ad-hoc collaborations. Rotaru *et al* propose a access control framework for GCS [12]. In this work, group administrators can add or modify assignments to meet the application and local administration requirements without global administrators' involvement. It is a flexible administration mechanism for dynamic user-role assignment. However, they fail to provide practical access control architecture and implementation mechanism in GCS. Also, they do not consider security in ad-hoc collaborations between different groups, which is one of the main motivations by using GCS.

III. Group-based RBAC Model for Secure Collaborations

A. GB-RBAC Model

GB-RBAC incorporates the component of groups into traditional RBAC model. As RBAC96 model [18] and its administrative models have been extensively studied in literature, we build our model based on it. Figure 1 shows the components of GB-RBAC. The concepts of users (U), roles (R), role hierarchy (RH), permissions (P), permission-role assignment (PA), and sessions (S) are identical to those in RBAC96. Particularly, a role is assigned with a set of permissions and a user is assigned to a set of roles, both by security officers or system administrators. In a particular access *session*, a user activates a subset of assigned roles and obtains the permissions assigned to these roles. Roles can form a partial order hierarchy such that a senior (or higher level) role inherits the permissions of its junior (or lower level) roles. By configuring permission-role and user-role assignments, many security objectives can be achieved efficiently, such as least of privilege and separation of duty. Constraints can be defined for sessions, role hierarchy, and user-role and permission-role assignments for fine-grain authorization controls, such as, a user cannot activate two conflict roles in a single session for dynamic separation of duty purpose.

Besides these, GB-RBAC includes a set of groups (G). Each group is associated with a set of roles (role-group mapping or RGM) and permissions (permission-group mapping or PGM). A user can be the memberships of one or more groups, which is represented as the user-group mapping (UGM). In addition, we propose two layers of roles and permissions, which are referred as system-level roles (SR) and permissions (SP), and group-level roles (GR) and permissions (GP), respectively. Typically, in a GB-RBAC system, a system-level role associates global permissions which manage system-level resources, while a group-level role associate permissions in a small scope, for example, a particular application with specific resources, or a temporal collaborative task between some users.

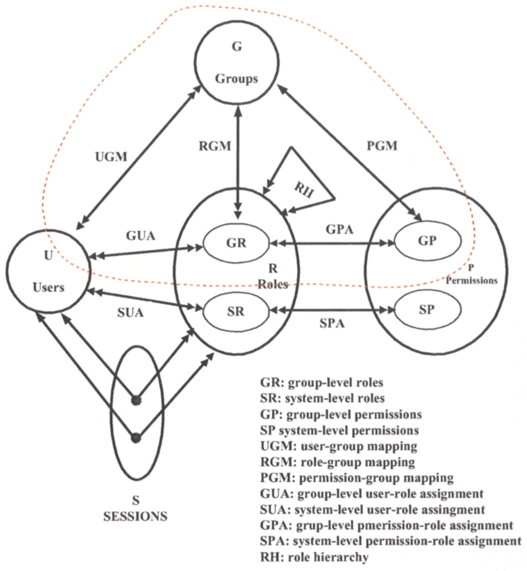

G
Groups
UGM
RGM
PGM
RH
GUA
GPA
U
Users
GR
R
Roles
GP
P
Permissions
SUA
SR
SPA
SP
S
SESSIONS
GR: group-level roles
SR: system-level roles
GP: group-level permissions
SP system-level permissions
UGM: user-group mapping
RGM: role-group mapping
PGM: permission-group mapping
GUA: group-level user-role assignment
SUA: system-level user-role assingment
GPA: grup-level pmerission-role assignment
SPA: system-level permission-role assignment
RH: role hierarchy

Fig. 1. GB-RBAC model

Besides the user-role assignment in system scope which is similar to the user-role assignment of ARBAC97 (URA97) [16], there is another type of user-role assignment which happens in group scope. Specifically, as UGM associates users with groups and RGM associates roles to groups, a group administrator can assign a user in the UGM to a role in the RGM, which is called group-level user-role assignment (GUA), while the original one is called system-level user-role assignment (SUA). In another word, GUA serves as the mechanism through which a role can be assigned to a user because the user *belongs* to a group and the role is defined within the group, and then the user holds the local permissions to access resources assigned with the role.

As a summary, a typical GB-RBAC model can be specified with the following components:

- U, SP, GP, SR, GR, S, and G (users, system-level permissions, group-level permissions, system-level roles, group-level roles, sessions, and groups, respectively).
- $P = SP \cup GP$, and $SP \cap GP = \emptyset$.
- $R = SR \cup GR$, and $SR \cap GR = \emptyset$.
- $UGM : U \rightarrow 2^G$, a function mapping a user to a set of groups. This relation shows that a user can be mapped into many different groups.
- $PGM : GP \rightarrow G$, a function mapping a group-level permission to a group. This relation shows that a group-level permission is identifyied within the context of a single group.
- $RGM : GR \rightarrow G$, a function mapping a group-level role to a group. This relation shows that a group-level role is defined within the context of a single group.

- $SUA \subseteq U \times SR$, system-level user-role assignment.
- $GUA \subseteq U \times GR$, group-level user-role assignment, and $(u,r) \in GUA$ only if $\exists g \in G, g \in UGM(u) \wedge g = RGM(r)$.
- $UA = SUA \cup GUA$, the overall user-role assignment relation.
- $SPA \subseteq P \times SR$, system-level permission-role assignment.
- $GPA \subseteq GP \times GR$, group-level permission-role assignment, and $(p,r) \in GPA$ only if $\exists g \in G, g = PGM(p) \wedge g = RGM(r)$.
- $PA = SPA \cup GPA$, the overall permission-role assignment relation.
- $RH \subseteq R \times R$, a partial order on R called the role hierarchy or role dominance relation. For any two roles r_1 and r_2, $r_1 \geq r_2$ means that r_1 has partial relation over r_2.
- $user : S \rightarrow U$, a function mapping each session s to a single user. $user(s)$ is constant within s.
- $permissions : R \rightarrow 2^P$, a function mapping a role to a set of assigned permissions, and $permissions(r) = \{p | \forall r' \leq r, (p, r') \in PA\}$.
- $roles : S \rightarrow 2^R$, a function mapping a session to a set of roles, and $roles(s) \subseteq \{r | (\exists r' \geq r)[(user(s), r') \in UA]\}$, which may change within session s, and session s has the permissions $\bigcup_{r \in roles(s)} \{p | (\exists r'' \leq r)[(p, r'') \in PA]\}$

Two levels of policy administrations exist in GB-RBAC. Particularly, there are system and group security administrators, which controls the SUA/SPA and GUA/GPA, respectively. Typically, system administrators can specify the set of users and permissions that a group administrator can manage, i.e., through the relations of UGM and PGM, respectively. Role-group mapping (RGM) can be defined by system administrators or group administrators, depending on applications. Through these, GB-RBAC achieves the separation of administrative privileges in different levels, as the dash line indicates in Figure 1.

In general, a user can be assigned to both group-level roles and system-level roles by different levels of administrators, although in a real system she may only be assigned to one level of roles. The set of assigned roles of a user includes those in SUA that are assigned by system administrators, and those in GUA that are assigned by group administrators. The user obtains all the permissions assigned to these roles through SPA and GPA, respectively. In GB-RBAC, a user can be assigned to a group-level role by local (group) administrators if the user belongs to the group, according to group-level authorization policies. In another word, GUA and GPA serve as the mechanism through which a role can be assigned to a user because the user belongs to a group and the role is defined based on the permissions within the group, and then the user holds the permissions to access resources defined with the group-level role.

Comparing with traditional RBAC models, GB-RBAC has the following unique benefits, simplified centralized user-role assignment for system administrators, flexible administration for dynamic environment, fine-grained user-role and permission-role assignment , and tunable group-level administration. With these advantages, we show how we apply GB-RBAC to build authorization scheme in GCS. Before this we introduce the concept of virtual group.

B. Virtual Group

Since collaborations frequently happen between groups in GCS, static access control policies can not control operations in collaborative jobs. In addition, as most group collaborations are dynamically performed by groups administrators in ad-hoc manner, it is impossible to predefine polices for all collaborative tasks. A desired security requirement in GCS is that each group has autonomy to control its own resources and permissions, thus global system administrators need not be involved. For this purpose, we introduce a special group – *virtual group* (VG) in GB-RBAC.

A VG has the similar features as common groups except that it is dynamically created corresponding to a collaborative task. Typically, a collaboration is initialized by a user of an existing group who has the corresponding permission, e.g., a group administrator, as this is a sensitive operation of the group. According to this, a VG is created with unique identity, and the initializing user can be the administrator of the VG, or a user is assigned by this administrator [1].

Based on collaborative tasks and security requirements, a set of permissions and roles can be defined by the VG's administrator. Specifically, the shared resources of the collaboration are regarded as objects, and permissions are defined based on different actions on these shared resources. Different collaborative applications have different types of shared resources. For example, in a video conferences among attendees from various groups, communication channels to all participants and between individuals are the shared resources. Permissions are operations to these resources, e.g., joining the conference, broadcast talk, private talk, controlling an attendee's talking time, uploading document for displaying, etc. Roles like host, reporter, and different level of speakers can be defined and assigned with appropriate permissions. In many other collaborations, shared resources can be contributed by individual groups, such as those in Grid computing environment [7]. In these cases, the permissions of a VG are defined by its resource providers.

One a VG is created, users from the initializing group and other collaborative groups can join this VG. Hereafter we call these collaborative groups *source groups*. The administrator of the VG assigns different roles to these users, according to their duty and tasks in the collaboration.

C. Automatic Policy Generation for Virtual Group

Security policies in a VG control the operations of groups members from source groups. As we use role-based access

[1]In general, more then one administrator of a group can be assigned, each with different administrative permissions. For simplicity, here we assume the uniform permission of all administrators of a group.

control model, the key issue is to assign users to appropriate roles. For ad-hoc collaborations among groups, it is tedious to have global system administrators to manage these. Further, administrators of a collaborative job has more context of dynamic user behavior. Thus it is very desirable that user-role assignment should be handled by local administrator within a group, and the procedure should be automatic or semi-automatic such that a user is assigned to roles once she joins the group or access group resources.

We provide an automatic policy generation mechanism to export users from source groups into a VG and assign roles. Similar mechanisms can be defined for permissions of a VG. However, as the way of sharing resources in a VG is application-specific, we focus on the user-role assignments. The automatic mechanism is triggered by group administrators when a collaboration is required and a VG is created.

As collaborations are built among existing groups, a user's job in a VG can be determined by her roles in source groups. We use the concept of *prerequisite conditions* to specify constraints that a user can or cannot be assigned to a role in a VG, based on her existing roles. A prerequisite condition is a logic rule that has to be followed when assigning users to roles by group administrators, which consists of predicates built on the relations of a GB-RBAC model. For example, a user prerequisite condition can test a user's memberships/nonmemberships of roles and groups, which is very useful in assigning users to roles in a VG by administrators. For an instance, for separation of duty (SoD) reason, the following rule indicates that for any user x, it can be assigned to role $host$ only if there is no other user y which is from the same group of x and assigned to role $host$ or $invited\_speaker$.

$$(x, host) \in UA \rightarrow \nexists y,\ UGM(x) \cup UGM(y) \neq \emptyset \wedge ((y, host) \in UA \vee (y, invited\_speaker) \in UA),$$

where x and y are two user variables. With this rule, any two users from the same group cannot be assigned to the $host$ and $invited\_speaker$ of a single VG. Note that here we use global role namespace, which can be augmented with group-identified namespaces. Other conditions can be defined similarly. For example, for least privilege purpose, a user cannot be assigned to roles out of her job duty. Also, cardinality constraints can be defined, such as a user cannot be assigned to more than one group during collaborations, or a role cannot be assigned to more than one user.

When all conditions have specified for a VG, the RBAC policy (group-level user-role assignment) can be dynamically generated upon user access requests. Specifically, when a user explicitly wants to join a group or access shared resources of a group, roles of the VG are assigned to the user based on her job in the VG. Conditions are checked before the roles are assigned, e.g., according to her roles in source groups, to determine whether the assignment can be allowed or not. We explain more details of this in next section with the context of collaboration in Spread.

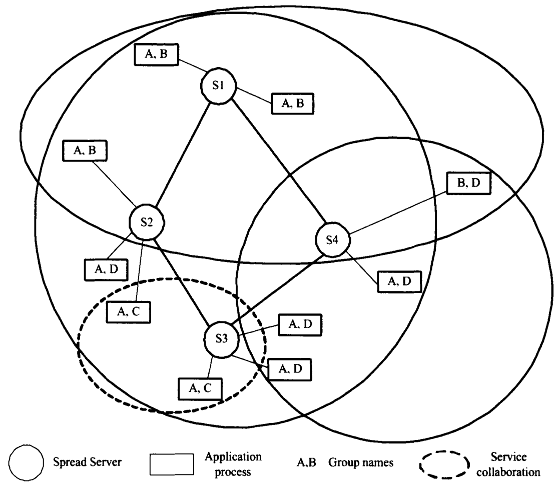


Fig. 2. Spread Architecture

IV. Secure GCS with Spread

A. Spread Architecture

Spread is a general-purpose GCS for wide and local area networks [5]. It provides reliable and ordered delivery of messages as well as a membership service. The system consists of a server and a client library linked with the application. This architecture amortizes the cost of expensive distributed protocols, since such protocols are executed only by a small number of servers. This way, all operations (including join and leave events) of a client process translates into single messages. Spread can be configured to use a single daemon in network or to use one daemon in every computer running group communication applications. Figure 2 presents a case where each computer executes one Spread daemon, by which all physical communications are handled. Specifically, Spread daemons keep track of computers' memberships. Each daemon keeps track of processes residing on its machine and participating in group communication. This information is shared between daemons to create the lightweight process group membership.

B. Access Control in Spread

Spread offers a many-to-many communication paradigm where any group member can be both a sender and a receiver. Although designed to support communication among many groups, it can accommodate a large number of collaborative sessions. Spread scales well with the number of groups used by the application without imposing any overhead on network routers. As we described above, in a Spread session, Spread servers have different types of protected resources (users and communication channels). Typically, there are five types of actions to the resources: *join*, a user joins a Spread session; *send*, a user in a group sends broadcast messages to all the members of the group; *p2psend*, a user in a group sends p2p messages to a special member in the group; *leave*, a user leaves a session she is involved; and *manage*, a user in a group who obtains the management rights manages all the sessions of the group [2]. In this way, we need to determine that a user in a group can perform which type of operations on which object.

For secure Spread, dynamic policy generation is also needed for secure group collaboration, since many different groups in Spread may join a third group for a new communication session. In this scenario, group administrators should have permissions to trigger dynamic policy generation to support ad-hoc collaboration based on the policies of involved groups. Our extended GB-RBAC model provides dynamic policy generation to support secure ad-hoc collaboration in Spread.

Permission check is enforced before an operation is executed. After the authorization, all the permission information are kept in the session. Every group has a set of policies to control communications among different group members in GCS from general groups and VGs. The goal of these policies is to reserve the final control of groups while provide management flexibility. In ad-hoc collaborations with GCS, there is no pre-established security policy such that user access control should be performed by individual groups. VGs are introduced to handle this situation, and group communication without collaboration is a special case of group communication in GCS. The following interactions show authorization workflow for a collaboration.

1) Before communicating with other groups, a VG G_v is created by the initializing group (say G_x) administrator if G_x builds collaboration with other groups (e.g., G_y and G_z). The administrator automatically becomes the administrator of G_v. The group administrators of G_y and G_z can be be G_v's administrators with negotiation between G_x.
2) Based on operations in group G_v by Spread, a set of roles R_v are defined by the administrator of G_v. That is, $(G_v, r) \in RGM$ only if $r \in R_v$.
3) Based on VG-level security requirements, G_v administrators assign roles to a user whenever she joins. This group-level user-role assignment can be performed before or upon access requests. For example, when joins G_v, a user is assigned with roles after authenticated.
4) For each assignment step (say a user from G_x), if there is a condition where G_x and any role in G_x or G_v is involved, then the condition is checked if the assignment can be allowed.
5) Once roles are assigned to users, the permissions of a user can be determined by G_v authorization service when access requests are generated from the user to perform operations in Spread.

On a high level view, in ad-hoc collaborations, each group has its own security administrators such that local users are authorized to perform operations in Spread. Note that administrator in groups can be implemented as authorization service such that the VG creation and user-role assignments can be automatically generated. We apply this approach in our prototype system.

[2]Original Spread package [5] does not provide this function, we extend it for management purpose in group collaborations.

V. Prototype Implementation and Experimental Study

To show the feasibility and performance of our framework, we implement a secure Spread prototype system, which enables different group of members to start secure collaborative communications.

A. Prototype Overview

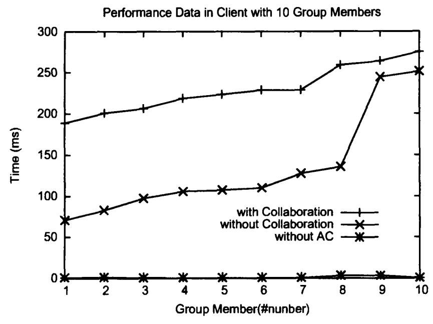
The Spread service in our prototype provides a platform for different groups to communicate collaboratively. The core building block of the service is the group communication system Spread [4]. The policy service is build based on Sun's XACML [20].

The prototype comprises three applications, one running on each machine, implemented as authorization server, Spread server and client platform respectively. The Spread server is built on a Linux-2.6.12 machine which has Pentium 4 1.7 GHz CPU and 640MB memory, and uses Spread 3.0. The authorization server is in Java 1.4.2 and working in Windows XP machine which has Pentium M 1.7 GHz and 512MB memory. The user platform used in the prototype system is built on a Fedora-2.6.9 machine which has Centrino 1.3GHz CPU and 512MB memory, which simulates to generate concurrent access requests.

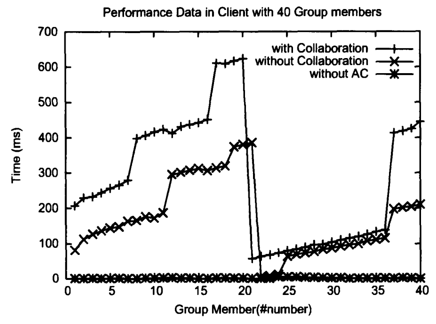
B. Performance Evaluation

As an access control decision is dynamically determined based on the requesting subject, the target object, the required action, and the group that the subject belongs to, the performance of the system should be considered. Because the overhead of the system is introduced in user authorization, time variation for all type of operations in Spread are same. We only discuss the performance data of join event in the following three cases: no access control, access control without group collaboration event, and access control with group collaboration event.

The performance results of join events in user platform are presented in Figure 3(a) and 3(b) with concurrent 10 join events and 40 join events, respectively. The figures show that the time in different types of join events (with collaboration, without collaboration and without access control (AC)) is increasing with more concurrent join events. However, the time increase (actually the "saw-like" behavior) in different join events shown in Figure 3 is not caused by access control enforced in the concurrent join events, but by TGDH-based session key negotiation during join events [2]. The similar results are also seen when no AC is enforced (illustrated in Figure 4. The largest process time of join events with collaboration is about 620 $msec$ and the largest process time of join events without collaboration is about 380 $msec$. As the policy process time is very small (less than 100 μsec) and permission query in Spread server is stable (about 10 $msec$), thus the main overhead is introduced by the socket connections among three different platforms. Since AC is only checked once during a type of continuous operations, we believe that the authorization performance is acceptable for GCS.



(a) Performance in client with 10 join events



(b) Performance in client with 40 join events

Fig. 3. Performance Results with concurrent join events

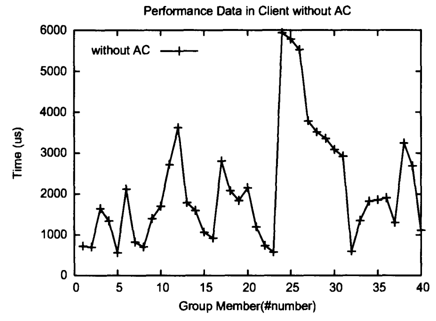


Fig. 4. Performance result in client without AC

VI. Conclusion

This paper presents an authorization architecture for secure group communication system. In particular, we propose a collaborative GB-RBAC model for GCS toward efficient authorization. Our model supports fine-grained and scalable access control for collaborations in GCS with easy policy management. Virtual group (VG) in our model enables the mechanism of on-demand policy generation and is the base for collaborative group communication. In addition, we show how our model could be integrated into Spread architecture and present experimental results that offer insights into its scalability and practicality.

Acknowledgment

We would like to thank Seth Proctor and Anne Anderson from Sun Microsystems for their valuable suggestions to improve the prototype of the authorization server. Especially, we are grateful to Prof. Yair Amir from Johns Hopkins University for his helpful discussion about the Spread performance. This work is supported by the Natural Science Foundation of China (No. 90604024), the Key Project of Chinese Ministry of Education (No. 106012), NCET and HI-Tech Research and Development Program of China (863) (2007AA01Z2A2).

References

[1] R. Alfieri, R. Cecchinib, V. Ciaschinic, L. dell'Agnellod, A. Frohnere, K. Lorenteyf, and F. Spatarog. From gridmap-file to voms: Managing authorization in a grid environment. *Future Generation Computer Systems 21*, 2005.

[2] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key agreement protocols. *ACM Transactions on Information Systems Security*, 7(3):257–488, August 2004.

[3] Y. Amir, C. Nita-Rotaru, and J. Stanton. Framework for authentication and access control of client-server group communication systems. In *Proceedings of the Third International Workshop on Networked Group Communication*, 2001.

[4] Y. Amir, C. Nita-Rotaru, J. Stanton, and G. Tsudik. Secure spread: An integrated architecture for secure group communication. *IEEE Transactions on Dependable and Secure Computing*, 2(3):248–261, July 2005.

[5] Y. Amir and J. Stanton. The spread wide area group communication system. *Tech. Rep. 98-4, Johns Hopkins University, Center of Networking and Distributed Systems*, 1998.

[6] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. Richard Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3), 2001.

[7] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115-128, 1997.

[8] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[9] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *ACM Conference on Computer and Communications Security*, pages 235–244, 2000.

[10] M. Lorch, D. B. Adams, D. Kafura, M. S. R. Koneni, A. Rathi, and S. Shah. The prima system for privildge management, authorization and enforcement in grid environments. In *Proceedings of the 4th International Workshop on Grid Computing*, 2003.

[11] P. McDaniel, A. Prakash, and P. Honeyman. In *Proceedings of the 8th USENIX Security Symposium*, 1999.

[12] C. Nita-Rotaru and N. Li. A framework for role-based access control in group communication systems. In *Proceedings of International Workshop on Security and Parallel and Distributed Systems*, 2004.

[13] S. Oh, R. Sandhu, and X. Zhang. An effective role administration model using organization structure. *ACM Transactions on Information and System Security*, 9(2):113-137, May 2006.

[14] L. Pearlman, V. Welch, I. Foster, and K. Kesselman. A community authorization service for group collaboration. In *Proceedings of IEEE Workshop on Policies for Distributed Systems and Networks*, 2002.

[15] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. *ACM Computing Survey*, 35(3):309–329, 2003.

[16] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of role. *ACM Transactions on Information and Systems Security*, 2(1):105–135, February 1999.

[17] R. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and Systems Security*, 2, 1999.

[18] R. Sandhu, E. Coyne, H. Reinstein, , and C.Youman. Role-based access control model. *IEEE Computer*, 29(2):38–47, February 1996.

[19] Y. Sun and K. J. Ray Liu. Scalable hierarchical access control in secure group communications. In *INFOCOM*, 2004.

[20] Sun's XACML. http://sunxacml.sourceforge.net/.