Towards a Group-based RBAC Model and Decentralized User-Role Administration

Qi Li [1,2], Mingwei Xu [1,2], Xinwen Zhang [3]

[1]*Department of Computer Science, Tsinghua University, Beijing 100084, China*
[2]*Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China*
[3]*Samsung Information Systems America, San Jose, CA, 95134, USA*
{liqi,xmw}@csnet1.cs.tsinghua.edu.cn; xinwen.z@samsung.com

***Abstract*— Role-based Access Control (RBAC) has been widely deployed in many distributed systems in recent years. However, in many large-scale enterprise environments, it is difficult to manage RBAC because of the huge number of users and roles, and complex interrelationships between them. Moreover, with the development of information and communication technologies, many temporal and ad hoc collaborations between groups and departments are emerging, which require dynamic user-role and permission-role assignments. In these scenarios it is infeasible, if not impossible, for few security officers to administrate the assignment for various applications. In this paper, we propose a novel RBAC model for decentralized and distributed systems. As one of the main contributions, we also propose a decentralized administration model to address the management issues in traditional RBAC systems, Our model can be used for group-based applications with dynamic assignments where typically local (group-level) administrators take charge of the dynamic assignments. In this way, many administrative tasks for different applications can spread over many different local administrators, and a fine-grained administration model of RBAC based on the local administration policies is realized. As a proof-of-concept system, we implemented a secure Spread prototype based on our proposed model to show the feasibility in the real applications.**

I. Introduction

The past decade has been seeing the emergence and wide use of many e-commerce and e-government systems, Internet-based services are becoming more and more important. Especially, with the development of information and communication technologies, many temporal and ad hoc collaborations between groups and departments are emerging. To widely deploy these systems, an important concern is the information security problem. To give a practical solution, researchers have designed and implemented many access control systems. Among them, Role-based Access Control (RBAC) is the most attractive solution with the property of "policy-neutral" in the sense that by using roles and constraints, a wide range of security policies can be expressed, including Discretionary Access Control (DAC), Mandatory Access Control (MAC), and User-specific Access Control (USAC) [2], [5]. Because the numbers of roles and users in an RBAC system vary from tens to thousands in large enterprise systems, the management becomes an important issue. Many researchers provide various administrative models for RBAC to provide administration convenience and management efficiency [4], [3], [11], [14], [9]. Besides these models, some reference models for RBAC are proposed, such as the role graph model to implement the administration of role-role hierarchy [8] and the group graph model to provide group-role assignment [10].

However, these administration models mentioned above share the same weaknesses: (1) it is difficult to manage RBAC components, especially with hundreds or thousands of users, roles, and their interrelationships, since in general only a small team of security officers and individual department administrators are delegated to manage these components; (2) the problem becomes worse with dynamic user-role and permission-role assignments in large-scale enterprise environments, where various group-based applications and systems emerge [7]. For example, different video conferences always need different users in a group to participate, so there must be many administrative tasks for administrators to dynamically modify role and permission assignments for video conferences. It is impossible and infeasible for few local administrators to administrate these assignments. Therefore, the management issue of RBAC is an important factor which directly restricts its deployment and usage.

Although some previous research work use partly decentralized administration for RBAC, they are not flexible to use in dynamic assignment environments. The main point is that the administration is still under the control of an administration domain which consists of a few administrators. Our motivation behind this issue is to simplify the administration thus enhance the practicability of RBAC in dynamic environments. In this paper, we propose a truly decentralized and group-based RBAC (GB-RBAC) model by introducing the concept of groups and modifying the user-role assignment model based on Sandhu's work [9], [14]. The GB-RBAC model retains the main features of RBAC, and adds a default user-role assignment which is enabled without administrator's involvement. At the same time, a group administrator can assign other explicit roles to a group member based on local administrative policies in a fine-grained manner instead of system administrators. Hence, fine-grained user-role assignment is proposed in our model. Based on our advanced model, we provide a new user authorization mechanism and a two-level administration model to facilitate RBAC administrative issues. In this way, our model supports decentralized management in a simple and efficient way. For example, in group collaboration systems [7], 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. Moreover, the decentralized administration model we proposed provides tunable group-level administration controlled by system-level administration. Therefore, not only is user-role assignment for system administrator greatly simplified, but the principle of the duty of separation is also embodied. With these features, our administration model satisfies the requirement of autonomy administration for RBAC, which is not solved completely by previous work because of centralized administrative domain with few system-level administrators, and thus supports ad hoc collaboration between different departments/groups. In addition, we develop an authorization framework based on the GB-RBAC model and the prototype of the model shows the feasibility in the real distributed applications.

This paper is organized as follows. Section 2 summarizes previous work in this field and the difference from our approach. Section 3 introduces our proposed GB-RBAC model. Section 4 presents the administrative strategy which focuses on user-role assignment aspect. The implemented prototype system with proposed scheme is

1545-0678/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDCS.Workshops.2008.26

described in Section 5. Section 6 concludes this paper.

II. Related Work

Sandhu *et al* define a set of RBAC models [15], [14]. Afterwards, Ferraiolo *et al* propose the NIST standard of RBAC [6]. Different architectures for RBAC services on the web are proposed in [12]. In these RBAC models, however, static user-role association is used, which is tedious to configure every user-role assignment. Moreover, in the user-role administration model known as URA97 of ARBAC97 [14], multiple steps are needed to assign a user to a role because the prerequisite conditions of user-role assignment relations in URA97 are defined with regular roles, which form a role hierarchy [14]. As a result, it is difficult to administrate the policies when the model is deployed in a large enterprise system because it has many groups or departments. Our GB-RBAC provides a novel user authorization mechanism though $DSet$ and a group-level administration model. In this way, we do not need many steps to assign many roles to a user, thus effectively reduce the steps of assignments and duplicated user-role information. Moreover, our administration model easily supports another fine-grained user-role assignment besides previous administration model, because some parts of administration tasks can be performed by group administrators based on the local administration policies.

ARBAC02 [9] addresses the multiple user-role assignments and duplicated information problems by defining user-role relations based on existing organization structure information as user pools and permission pools, such as user's position from human resource department and permissions from IT department, instead of the regular roles. The limitation of this model is that roles in a user pool must have partial order relation. It results great constraints on user-role assignment in a situation where there are many diverse discrete roles. Another weakness in ARBAC'02 is that it requires predefined user pools (OS-U) and permission pools (OS-P), and this introduces complex tasks for administrators. To solve these problems, we add the concept of groups into the existing RBAC model. A group can be registered into proper role set and a user can be assigned to roles from the role set. As a result, there are less complicated requirements and user-role assignment becomes simpler. Because of this, our approach significantly improves the administration of the system, especially in large enterprise and web-based applications.

III. The GB-RBAC Model

This section first provides the overview of our proposed model, and then presents a formal description.

A. Overview of GB-RBAC

The GB-RBAC model proposed in this paper introduces the concept of group, through which user-role assignment is implicitly and explicitly conducted, which overcomes the weakness in most previous work aforementioned. The essential difference between groups and roles is that a group is a collection of users who have the similar security attributes, while a role is a collection of permissions[1] [13].

Figure 1 demonstrates all the components in GB-RBAC model and how it works. Besides those in previous RBAC model, GB-RBAC introduce two new components: groups and group leaders (administrators). A group, as mentioned above, is a collection of users and a group administrator is a user with the group administrative role, with which GB-RBAC model supports decentralized

[1]Although a role is considered as a set of users and permissions in RBAC [13], for user-role administration purpose, we consider a role as a set of permissions here.

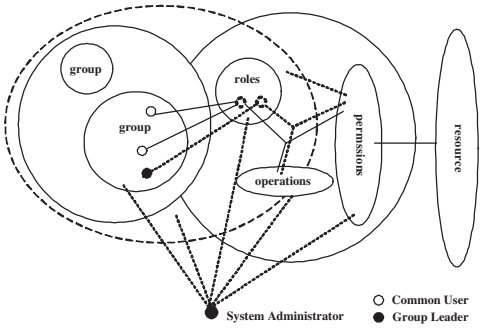


Fig. 1. Overview of a GB-RBAC system

administrations. In Figure 1, there are two types of lines: the solid lines denote data information flow and the dashed lines denote management information flow. The dashed lines among the groups, roles, permissions, and operations are also not seen in previous models. Actually these dashed lines illustrate a type of administration model. As shown in Figure 1, two levels of user administrations can be provided in a GB-RBAC system. The first is the system-level administration associated with centralized control over user-role assignment, that is denotes by the dashed lines whose start point is the node of the system administrator. The second is the group level administration associated with decentralized control over user-role assignment, which is denoted by the dashed lines whose start point is the node of group leaders. In the rest of this paper, the two terms, system-level administration model and group-level administration model, refer respectively to these two administrative approaches.

B. Model Description

GB-RBAC indirectly imposes access control on a user's action after this user is authenticated and assigned to a set of roles implicitly or explicitly. Figure 2 shows the components in a GB-RBAC model. The concepts of users (U), roles (R), role hierarchy (RH), permissions (P), permission-role assignment (PA), and sessions (S) are identical to those in the original RBAC96 model [15]. Besides these, a GB-RBAC model includes a set of groups (G). Each group is assigned with a set of roles (group-role assignment or GA). A user can belong to one or more groups, which is represented as the user-group mapping (UM).

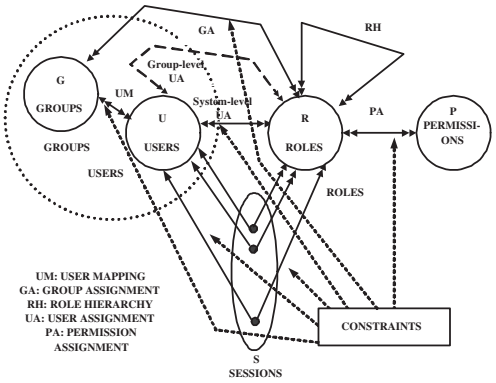


Fig. 2. GB-RBAC model

Besides the user-role assignment in system scope which is similar

to the user-role assignment in URA97 [14], there is another type of user-role assignment which happens in group scope. Specifically, as UM associates users with groups and GA associates roles to groups, a group administrator can assign a user in the UM to a role in the GA, 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 has a mapping relation with a group and the role is assigned to the group, and then the user holds the permissions to access resources defined with the role. Through the mechanism of mapping users to groups (UM) and assigning role to groups (GA), a new concept of default group roles set ($DSet$) is introduced in GB-RABC, which indicates the set of roles assigned to a group by default.

The formal definitions of individual components in GB-RBAC are presented in the remainder of this section.

Definition 1: A GB-RBAC model has the following components:

- U, P, R, S, and G (users, permissions, roles, sessions, and groups, respectively).
- $PA \subseteq P \times R$, a many-to-many permission to role assignment relation.
- $UM \subseteq U \times G$, a many-to-many user to group mapping relation. This relation shows that a user can be mapped into many different groups.
- $GA \subseteq G \times R$, a many-to-many group to role assignment relation.
- $SUA \subseteq U \times R$, system-level user-role assignment.
- $GUA \subseteq U \times R$, group-level user-role assignment, and $(u, r) \in GUA$ only if there is a group g and $(u, g) \in UM \wedge (g, r) \in GA$.
- $UA = SUA \cup GUA$, a many-to-many user-role assignment relation.
- $user : S \rightarrow U$, 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.
- $RH \subseteq R \times R$, 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.
- $roles : S \rightarrow 2^R$, 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 $\bigcup_{r \in roles(s)} \{p | (\exists r'' \leq r)[(p, r'') \in PA]\}$

Similar to that in RBAC96, GB-RBAC does not require each role to be assigned to at least one permission and each user to be assigned to at least one role. Also, GB-RBAC does not require each user to be mapped to at least one group.

Through the concept of group in Definition 1, we introduce the concept of default group role set.

Definition 2: The default role set of a group $DSet : G \rightarrow 2^R$ is a subset of R, and $\forall u, r, g, (u, g) \in UM \wedge r \in DSet(g) \rightarrow (u, r) \in GUA$. That is, a user who is mapped to a group obtains all the roles in the default role set of the group automatically.

The procedure to determine the permissions of a user in GB-RBAC is described as follows. When a user logins a system or starts an application, a session is created and a subset of the assigned roles of the user is activated. The set of assigned roles of a user includes the user's directly assigned roles (through SUA), the roles in the $DSet$ of the group that the user is registered, and the roles that are assigned by group-level administrators (through GUA). The user obtains all the permissions assigned to these roles through PA. Users can also change the activated roles in a session within his assigned roles. The session can be terminated by the user or by the system, e.g., because of a long idle duration. For simplicity in this paper we assume that in a single session a user cannot change his group membership.

From the formal description of the model, we can see that the use of roles in $DSet$ of a group does not take effect when the roles in the set are defined and there is no permission assigned to them. In addition, as each user in the group is assigned to a set of roles by default, user-role assignment can be changed by the administrative roles including the system administrative users and group administrative users. The detailed administration model is described in Section 4.

IV. GB-RBAC Administration Model

This section first gives an overview of user-role administration in GB-RBAC, then describes the formal model, and then discusses its advantages over traditional administration models. Due to space limit we focus on the use-role assignment while revocation is not included in this paper.

A. Overview

The success of an access control system is heavily dependent upon its administration, especially when the number of users and roles are on a scale of thousands. Managing the access control components and their inter-relationships is an important and formidable task. Compared with previous RBAC models, our model introduces some extra administration tasks, such as group-role assignment and user-group mapping. The administration model of user-group mapping (UM) is to classify the users into different groups and it is the duty of the system-level administration model. The group-role assignment (GA) is a novel mechanism for RBAC and this assignment is similar to the user-role assignment to some extent, which is in charged by system-level administrators. Besides these, the $DSet$ of a group is another administrative task that can affect the permission propagation in the model. The management of $DSet$ of a group can be implemented in both administration levels. However, we consider it in group-level administration since one of the motivation of the model is to provide group-level autonomy administration, such that a group administrator has the permission to assign users to the roles in GA relation.

For these two administration levels, two types of administrative roles are defined in the administration model, called system-level administrative roles (SAR) and group-level administrative roles (GAR). These administrative roles also can form role hierarchies, respectively, similar to that of the regular roles in GB-RBAC. For simplicity we assume that $SAR \cap GAR = \emptyset$. A user of a system administrative role (or simply, a system administrator) can assign a user to a group (through UM), but a user of a group administrative role (or simply, a group administrator) can determine which role the user can be assigned to. In this way, a type of separation of duty in different levels of administration is provided. In addition, UM can be managed by a administration model simpler than GA, and it does not add much complexity into the administration model of GB-RBAC.

As Figure 3 shows, two level administration models respectively referred as system-level and group-level administration model are proposed to address all kinds of components defined in the GB-RBAC definitions. Our administration model in this paper focuses on user-role assignment, so the administration model proposed in this paper will address these components in GB-RBAC: GA, UM, SUA, GUA, and DSet.

Specifically, system-level administration model has three types of administrative controls enforced on GA, UM, SUA , respectively,

while group-level administration model has two types of administrative controls enforced on GUA and $DSet$, respectively. Note that we do not address the administration of UA since it can be implemented by the administration of SUA and GUA through the Definition 1. Also note that as the controls on the $DSet$ of a group defines the default role set of the group members, it implicitly manages the user-role assignment since a user of the group has the memberships of the roles in $DSet$ automatically. For example, in a temporal group telephone conference, a role with common permission of connecting the virtual conference room is defined in the $DSet$, which is assigned to all users in the group by default. As a group's $DSet$ is very application-specific, we do not explicitly define the rules to manage $DSet$ in this paper.

Different layers of administration controls provide administration autonomy mechanism such that local administrators of a group can assign a member of the group to some different roles if some assignment conditions are satisfied (introduced shortly). It is another flexible mechanism to administrate UA besides the mechanism provided by $DSet$ to deduce the administration tasks in first level administration model.

Different level administration models have different responsibilities: administration of GB-RBAC components with centralized control over users is in the system-level administration model, and administration in the group view is in the group-level administration model. Figure 4 shows the scope of these two-level administrations. The system level model operates in system scope to assign/revoke roles to/from users, assign/revoke roles to/from groups, and map/unmap user to/from groups, while the group level model operates in group scope to assign/revoke roles to/from users, including defining roles in $DSet$. For sake of simplicity we do not consider permission-role and role-role assignments in this paper.

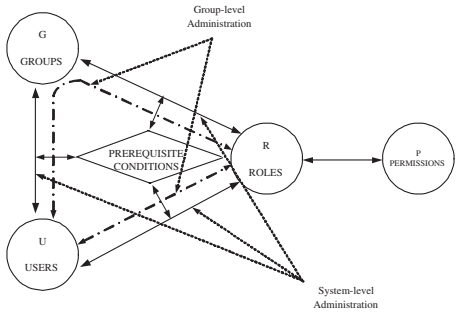


Fig. 3. GB-RBAC Administration Model

B. GB-RBAC Grant Model

1) User and Group Prerequisite Conditions: The notion of prerequisite condition is a key concept of our decentralized administrative model. Figure 4 illustrate the components of prerequisite conditions that can be defined.

Definition 3: A *user prerequisite condition* is defined as a boolean expression using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\overline{x}$, where x is a regular role (i.e., $x \in R$) or a group (i.g., $x \in G$). A prerequisite condition is evaluated for a user u by interpreting x to be true if any of the follows is true:

- if $x \in R, \exists x' \geq x, (u, x') \in UA$;
- if $x \in G, (u, x) \in UM$.

and interpreting $\overline{x}$ to be true if any of the follows is true:

- if $x \in R, \forall x' \geq x, (u, x') \notin UA$;
- if $x \in G, (u, x) \notin UM$.

For a given set of roles R and G, let CR_u denote all possible user prerequisite conditions that can be formed.

A user prerequisite condition can test a user's membership of role(s) and group(s). Note that a user's membership of a group is more than the combination of his memberships of the roles in the $DSet$ of the group, as a group administrator can assign the user to some other roles in the group. Thus, the test of group membership not only provides simplified administration but also an abstract of the user's role in group level.

As the membership of a role tests both SUA and GUA, the prerequisite condition defined above is at least as expressive as that in URA97 [14].

Definition 4: A *group prerequisite condition* is defined as a boolean expression using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\overline{x}$, where x is a regular role (i.e., $x \in R$). A prerequisite condition is evaluated for a group g by interpreting x to be true if $\exists x' \geq x, (g, x') \in GA$, and interpreting $\overline{x}$ to be true if $\forall x' \geq x, (g, x') \notin GA$. For a given set of roles R, let CR_g denote all possible group prerequisite conditions that can be formed.

A group prerequisite condition checks the GA relation to test the memberships/nonm-emberships of a group in roles, which is used in the administration of group-role assignment.

2) User-role Assignment: Authorizations on user-role assignment in these two levels are controlled by the following rules.

Definition 5: In system-level administration model,

- user-role assignment in SUA is controlled by means of the relation $can\_assign\_SUA \subseteq SAR \times CR_u \times 2^R$.
- user-group mapping in UM is controlled by means of the relation $can\_assign\_UM \subseteq SAR \times CR_u \times 2^G$.
- group-role assignment in GA is controlled by means of the relation $can\_assign\_GA \subseteq SAR \times CR_g \times 2^R$.

Definition 6: The user-role assignment in GUA is controlled by the relation $can\_assign\_GUA \subseteq GAR \times CR_u \times 2^R$.

Specifically, a relation $can\_assign\_SUA(x, y, \{z\})$ or $can\_assign\_GUA(x, y, \{z\})$ means that a member of the system or group administrative role x (or an administrative role senior to x) can assign a user to be a member of a role in role range $\{z\}$ if the user satisfies the prerequisite condition y; a relation $can\_assign\_UM(x, y, \{z\})$ means that a member of the system administrative role x (or an administrative role senior to x) can assign a user to be a member of a group in $\{z\}$ if the user satisfies the prerequisite conditions y; and a relation $can\_assign\_GA(x, y, \{z\})$ means that a member of the system administrative role x (or an administrative role senior to x) can assign a group to a role in role range $\{z\}$ if the group satisfies the prerequisite conditions y.

Suppose a short-term project is launched in an organization and a temporal group (PRO1) is created to develop it. Corresponding roles are created as shown in the left circle in Figure 4, and group and system administrative role hierarchy in Figure 5. We explain how to assign users to the roles to fulfill the project development by system and group administrators collaboratively. A set of administration rules are defined in the organization as Table I shows. We put an '@' in front of the group names to distinguish with role names.

Consider Alice is a member of the system administrative role E-SSO, and Bob is a member of the role ED. According to rule $can\_assign\_UM(E\text{-}SSO, ED, \{@PRO1\})$, Alice can assign Bob to group PRO1. At the same time, according to the rule can_assign_GA $(E\text{-}SSO, , [ER1, PL1])$, Alice also can assign any roles between ER1 and PL1 to group PRO1 since there is no condition

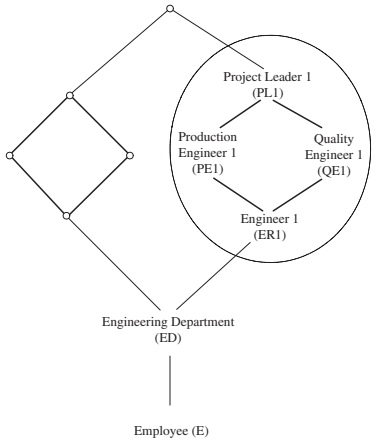


Fig. 4. Example role hierarchy

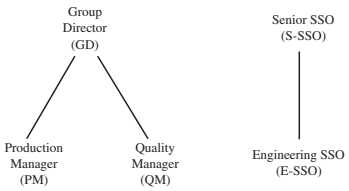


Fig. 5. Example administrative role hierarchy

specified. By assigning a role to a group, Alice enables that an administrative role in the group can assign users to this role, according to the local (group-level) policies defined as can_assign_GUA rules. In this example, suppose Alice assigns ER1, PE1, QE1, and PL1 to PRO1, and Carol is a member of PM in PRO1. According to $can\_assign\ \_GUA\ (PM,\ @PRO1 \wedge \overline{QE1},\ \{PE1\})$, Carol can assign Bob to PE1 if Bob is not a member of QE1; and according to $can\_assign\_GUA(GD, @PRO1, PL1)$, if Carol is also a member of GD in PRO1, then Bob can be assigned to PL1 by Carol. Note that Alice can selectively assign roles to PRO1 in the role range, or revoke the group-role assignment as discussed shortly. Thus an adjustable group-level role range is achieved with our model.

An assumption in the administration model is that a system administrator is trusted not to assign roles to conflicting groups. For example, in above case, if role range [ER1, PL1] has been assigned to PRO1, Alice and other administrators should not assign any of the role in this range to other group [2] (say PRO2), otherwise a user from PRO2 can have permissions in PRO1, e.g., to read/write sensitive data, which is not allowed in general. This assumption is also used in traditional RBAC administration models, e.g., administrators are trusted not to assign two conflict roles to a single user.

Note that the example rules defined here is not a complete set of rules to admin a system. For simplicity we ignore some administration components such as to assign ER1 in the $DSet$ of PRO1.

C. Advantages of Two-level Administration Model

We summary the main advantages of our administration model by comparing it with ARBAC97 as follows.

[2]In this paper we do not consider group hierarchy in the model. If a group belongs to another group, then a role can be assigned to both of them.

Type	Admin. Role	Rrereq. Condition	Group /Role Range
can_assign_UM	E-SSO	ED	{@PRO1}
can_assign_GA	E-SSO		[ER1, PL1]
can_assign_GUA	PM	@PRO1 $\wedge$ $\overline{QE1}$	{PE1}
can_assign_GUA	QM	@PRO1 $\wedge$ $\overline{PE1}$	{QE1}
can_assign_GUA	GD	@PRO1	{PL1}

TABLE I
EXAMPLES OF ADMINISTRATION CONTROL RULES

1) *Simplified User-role Assignment for System Administrators* In our model, an administrator only needs to specify the role range of a group through GA relations. After that, group administrators take charge of the user-role assignment in this local role range. This significantly simplifies the management task by delegating administrative permissions from centralized system-level administrators to decentralized group-level administrators, especially for dynamic and ad hoc collaborative applications.
2) *Flexible Administration for Dynamic User-role Assignment* Group-level administration can flexibly support dynamic user participation in group-based applications, as group-level administrators typically can easily obtain user activities. With pure system-level administration like that in traditional RBAC models, it is tedious to manage user-role assignment in dynamic environments.
3) *Fine-grained User-role Assignment* By enabling GUA, our model supports fine-grained user-role assignment in group level. Typically, a group administrator has more contextual information about local security requirements in the group and the users' skills, thus user-role assignment in this level provides fine-grained and context-aware authorization.
4) *Tunable Group-level Administrations* A system-level administrator can change the role assignment of a group, thus change the roles that a group administrator can assign users to. This greatly provides flexible and controlled group-level administrative permissions.

V. PROTOTYPE AND PERFORMANCE EVALUATION

To show the feasibility and performance of our GB-RBAC model, we implement a secure Spread prototype system, which enables different group of members to start secure communication with others.

The access control mechanism in Spread controls which user in a group is able to communicate with other members of the group in which way. Typically, the enhanced secure Spread provides the following functions:

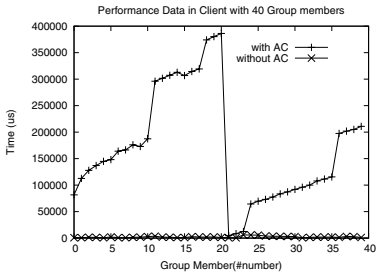
- A management module is implemented in the context of groups in Spread, and administrators (hosts) of a group can manage the session, e.g. a administrator can drop a group member's session dynamically.
- A access control framework is realized to control users' communication. In addition, in order to improve performance of the access control mechanism, all the permission information is kept in users' session after users are authorized.

The Spread service in our prototype provides a platform from different groups to communicate collaboratively. The core building block of the service is the group communication system *Spread* [1]. The prototype architecture is similar to the general architecture proposed in previous section. Specially, the policy service is built based on Sun's XACML [16].

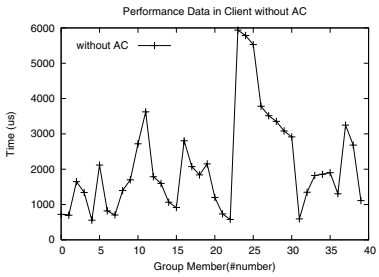
Three machines are involved in the experiments: Spread server, authorization server and client platform. The Spread server is builded on a Linux-2.6.12 machine which has Pentium IV 1.7 GHz CPU and 640MB memory, and uses Spread 3.0, and the authorization server are in Java 1.4.2 and working in Windows 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-2.5.9 machine which has Centrino 1.3GHz CPU and 512MB memory. The message are transferred through Socket among these three machines.

As a GB-RBAC decision is dynamically determined by subject, object, action attributes and group that subjects belong to, the performance of the system should be considered. Since the overhead of the system is introduced in first user authorization, time variation for all type the operations in Spread are same. So we only discussed the performance of join events.

The performance of join events in user platform is presented in Figure 6(a) which illustrates the performance with concurrent 40 join events. The process time in different types of join events, with access control (AC) and without AC, is increasing with more concurrent join events (The time increase in join events is caused by different join place in key tree). The largest process time of join event with AC is about 380 msec, and the average is about 68 msec. Since the time for policy process is very small (less than 100 usec)and permission query in Spread server is stable (about 10 msec), the main overhead is introduced by the socket connections among three different platforms[3]. In addition, time variety in different join events in Figure 6(a) is caused by TGDH-based session key negotiationand the time to join spread sessions also varies when no access control is enforced (illustrated in Figure 6(b)).



(a) Performance result with 40 join events



(b) Performance result without AC

Fig. 6. Performance results in user platform

[3]In real applications, the authorization server and Spread server may be deployed in one machine and the overhead will be greatly reduced.

VI. Conclusion

In this paper, we present an advanced RBAC model called GB-RBAC and its user-role administration. The main advantage of the model is convenience and flexibility for administration under large-scale environments. Our model does not need to be used in a highly central controlled environment, and provides two levels of administration models for user-role assignment and reduces the complexity of the administration of RBAC systems. Moreover, user-role assignment in the group-level administration model provides a flexible way to meet the requirements of group-level collaboration, i.e., users from different groups form a virtual group for communication. The prototype of the authorization framework based on GB-RBAC shows the feasibility in the real distributed applications. As the future work, based on the GB-RBAC model, small permission pools can be implemented in the group-level administration model.

Acknowledgment

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] 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.

[2] D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *Proceeding of IEEE symposium on Computer Security and Privacy*, pages 184–194, 1987.

[3] J. Crampton. Understanding and developing role-based administrative models. In *Proceedings of 12th ACM Conference on Computer and Communications Security*, pages 158–167, 2005.

[4] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and Systems Security*, 6(2):201–231, May 2003.

[5] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of 15th National Computer Security Conference*, pages 241–248, 1992.

[6] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and Systems Security*, 4(3):224–274, August 2001.

[7] 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.

[8] M. Nyanchama and S. Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and Systems Security*, 2(1):3–33, February 1999.

[9] 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.

[10] S. Osborn and Y. Guo. Modeling users in role-based access control. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, pages 31–38, 2000.

[11] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control policies. *ACM Transactions on Information and Systems Security*, 3(2):85–106, May 2000.

[12] J. Park, R. Sandhu, and G.J. Ahn. Role-based access control on the web. *ACM Transactions on Information and Systems Security*, 4(1):37–71, February 2001.

[13] R. Sandhu. Role versus group. In *Proceeding of 1st ACM Workshop on Role-Based Access Control*, pages 1–12, 1995.

[14] 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.

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

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