Atom-Role-Based Access Control Model

Weihong CAI††, Richeng HUANG††, Xiaoli HOU††, Nonmembers, Gang WEI†, Member, Shui XIAO††, and Yindong CHEN††, Nonmembers

SUMMARY Role-based access control (RBAC) model has been widely recognized as an efficient access control model and becomes a hot research topic of information security at present. However, in the large-scale enterprise application environments, the traditional RBAC model based on the role hierarchy has the following deficiencies: Firstly, it is unable to reflect the role relationships in complicated cases effectively, which does not accord with practical applications. Secondly, the senior role unconditionally inherits all permissions of the junior role, thus if a user is under the supervisor role, he may accumulate all permissions, and this easily causes the abuse of permission and violates the least privilege principle, which is one of the main security principles. To deal with these problems, we, after analyzing permission types and role relationships, proposed the concept of atom role and built an atom-role-based access control model, called ATRBAC, by dividing the permission set of each regular role based on inheritance path relationships. Through the application-specific analysis, this model can well meet the access control requirements.

key words: access control, RBAC, atom role, ATRBAC

1. Introduction

In recent years, role-based access control model has been widely researched and applied in some database systems and operating systems [1], [2]. The most representative study is RBAC96 model defined by Ravi Sandhu et al. Based on this model, the RBAC unified model was published as the NIST RBAC in 2000 and adopted as ANSI/INCITS standard in 2004.

Role hierarchy, which can increase the administrative efficiency of the model, is a natural means for structuring roles to reflect an organization’s lines of authority and responsibility. It is one of the merits of RBAC model that induces role hierarchy to imitate the work relationship in the real world. However, the senior role can fully inherit all permissions of junior role and all permissions may be performed by the top senior role, since the traditional role hierarchy model is built on unlimited role inheritance, which could not reflect the complex role structures in most large-scale distributed systems, and may easily cause the abuse of permissions. For example, in an e-mail system, the right which permits a user to view private letters in junior role should not be inherited by senior role unconditionally.

RBAC2 [4], a member of the RBAC96 family, provides for the introduction of constraints, but it seems an unnecessary overhead to introduce constraints because of a feature of the model, rather than a feature of the organization that is the subject of the model [6].

The main contribution of this paper is the introduction of an Atom-Role-Based Model that not only has an atom role hierarchical structure, which enhance the flexibility in access control, but also can implement limited inheritance which effectively avoids the abuse of permissions. Based on RBAC model, this paper, firstly proposed the concept of atom role by dividing regular roles according to inheritance path relationships among their permissions, and finally constructed a new atom-role-based access control model (ATRBAC). By the model analysis, we can see that the model well meets the needs of practical applications.

The rest of this paper is organized as follows. The related works are introduced in Sect. 2. Section 3 defines the model in detail. Section 4 is a model analysis. Conclusions and future work are presented in Sect. 5.

2. Related Works

RBAC model was firstly proposed by Ferraiolo et al. in the early 1990s [3]. After an in-depth study of RBAC, Sandhu et al. developed RBAC96 model [4], which has been widely recognized for its capability to meet multi-level requirements in applications. However, in some situation, it violates the least privilege due to the characteristic that the senior role can inherit all permissions of junior role in role hierarchy. So it did not accurately reflect access control requirements of most organizations [5], [6]. To solve this problem, Sandhu proposed the notion of private role consisting of private right which can’t be inherited by all senior roles [4]. It would not be in accord with some actual needs that some sensitive rights can be inherited by part of senior roles but not all. For example, in a complex organization structure of Fig. 1, $r_4$ may only want $p_5$ rather than $p_6$ to be inherited by $r_7$ out of the need for privacy.

Sandhu et al. [7] put forward ARBAC97 (Administrative RBAC'97) model, which limited the scope of administrator’s management by using prerequisite condition. The probability that unreasonable permission chains appeared was reduced in role hierarchy, and to a certain extent, the model can prevent the private (sensitive) right of junior role from being inherited by senior role. However, the problem...
of duplicate administrative work came into being. Sejong Oh et al. [8] redefined the prerequisite condition based on organization structure and proposed ARBAC02 model, which was to some extent able to avoid administrative permission propagation, but it failed to give effective support to hierarchy administration in organization. An additional mechanism for administrating role hierarchy called role template was proposed by Essmayr et al. [9], which can set kinds of template roles according to different needs, but the introduction of template role made the structure of role hierarchy even more complex.

The solutions involved limited inheritance by the means of extending permissions [10], [11]. However, they could result in the problem of combination explosion and role inflation. Based on RBAC96 model, EHRBAC (Extended Hierarchy Role-Based Access Control) model was raised by Zhong et al. [12], which introduced normal inheritance and extended inheritance mechanism. The extended inheritance would not only inherit public permissions but also private permissions, which went against the security policy that self-governing branches could be allowed to have partial autonomy. Lee et al. [13] proposed RPI-RBAC (Restricted Permission Inheritance RBAC) model, which ensured partial inheritance by dividing each common role into multiple sub-roles. Although RPI-RBAC can reflect diverse role relationships to a certain extent and prevent private or sensitive right from being inherited by senior role unconditionally, the sub-role was obtained from the division of role based on job functions and ways of role inheritance, therefore, the model could not take the differences of permission types and properties into consideration.

3. ATRBAC Model

3.1 Basic Elements

According to the standard ANSI INCITS 359-2004 for RBAC model [14], the following basic sets, relations and operational symbols are used.

**Definition 1** RBAC model

*Users, Roles, Permissions, Sessions* for sets of users, regular roles, permissions, respectively.

**UA ⊆ Users × Roles**: the many-to-many mapping from users to regular roles.

**PA ⊆ Permissions × Roles**: the many-to-many mapping from permissions to regular roles.

\[ \sigma(r : Roles) \rightarrow 2^\text{Permissions} \]: the function mapping a regular role to its assigned permissions.

**Definition 2** regular role relations

In the complex role structure, the regular role hierarchy is not a mathematically partial order, since there are complete and incomplete inheritance relations among them. The regular role hierarchy is written as \( \geq \). If \( r_j \geq r_i \), it indicates that \( r_j \) is a senior role of regular role \( r_i \).

**Definition 3** atomic Inheritance

The inheritance from a junior role to one of its senior role is said to be an atomic inheritance if all permissions of the former can be inherited by the latter.

3.2 Categorization of Role’s Permission

A role can be defined as a set of responsibilities and actions associated with a particular working activity. The unconditional inheritance mechanism will enable senior roles to have all the permissions of junior roles, which has some practical limitations. For example, in an e-mail system, personal operation on a private letter should be regarded as a private right, therefore, system administrators should not inherit this permission unconditionally. Hence, we classify the permissions of regular role into three types based on the properties of permission and inheritance ways, viz. public permissions (PBP), private permissions (PRP), and selective permissions (PSP). In the following definitions, \( PBP : Roles \rightarrow 2^\text{Permissions} \), \( PRP(r) \) is a public permission set of regular role \( r \); \( PRP : Roles \rightarrow 2^\text{Permissions} \), \( PRP(r) \) denotes a private permission set; \( PSP : Roles \rightarrow 2^\text{Permissions} \), \( PSP(r) \) is a selective permission set of \( r \). For convenience, \( TPSP(r, p, r_j) \) and \( FPSP(r_j, p, r_i) \) are introduced. The former denotes that the selective permission \( p \) in \( r_j \) can be inherited by \( r_i \) while the latter is opposite, where \( r_j \geq r_i, p \in \text{PSP}(r_i) \). For instance, in Fig. 1, if \( p_3 \) is a public permission in \( r_5 \), it can be inherited by senior roles \( r_6, r_8 \) and \( r_9 \). If \( p_4 \) is a private permission, then it can’t be inherited by any senior roles. If \( p_4 \) is a selective permission, it will be the responsibility of security administrator to define which senior role can inherit permission \( p_4 \). E.g., \( TPSP(r_5, p_4, r_6) \), \( TPSP(r_3, p_4, r_5) \) and \( FPSP(r_5, p_4, r_6) \) have been defined by security administrator, it means that, \( p_4 \) can be inherited by \( r_6 \) but not \( r_8 \). For continuity of inheritance, security administrator can define \( TPSP(r_i, p, r_j) \), if and only if \( r_j \) is the direct senior role of \( r_i \), or \( TPSP(r_i, p, r_j) \) holds and \( r_j \) is the direct senior role of \( r_i \). Three permission types and their characteristics are shown in Table 1.

3.3 Role Relationships

By Definition 1, we can see that \( \sigma(r) = PBP(r) \cup PRP(r) \cup PSP(r) \). Three different role relationships are brought up to deal with different kinds of permissions and their character-
Table 1 Categorization of permission in the regular role.

<table>
<thead>
<tr>
<th>Role/Permissions</th>
<th>Inheritance ways</th>
<th>Characteristics of assigned permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Permissions (PBP)</td>
<td>unconditional inheritance</td>
<td>permissions can be inherited by senior roles. It will have the feature of unconditional inheritance, which means, to a senior role ( r_k ) of the regular role ( r ), all permissions in ( PBP(r) ) can be inherited by ( r ).</td>
</tr>
<tr>
<td>Private Permissions (PRP)</td>
<td>no inheritance</td>
<td>permissions can not be inherited by senior roles. It will have the feature of no inheritance, which means, to any senior role ( r ) of the regular role ( r_k ), all permissions in ( PRP(r) ) can not be inherited by ( r ).</td>
</tr>
<tr>
<td>Selective Permissions (PSP)</td>
<td>conditional inheritance</td>
<td>permissions may or may not be inherited by senior roles. The security administrator specifies which selective permission can be inherited by which senior role. ( TTPS(r_i,p,r_j) ): the selective permission ( p ) in ( r_i ) can be inherited by ( r_j ). ( TFPSP(r_i,p,r_j) ): the selective permission ( p ) in ( r_i ) can not be inherited by ( r_j ).</td>
</tr>
</tbody>
</table>

Fig. 2 Three kinds of the role relationship.

3.3.1 Parent-Child Relationship

**Definition 4** parent-child relationship (\( \rightarrow \))

Two roles \( r \) and \( r_j \) shall satisfy a parent-child relationship which is denoted by \( r_i \rightarrow r_j \), if and only if \( \sigma(r_i) \subseteq \sigma(r_j) \).

As shown in Fig. 2 (a), the parent role node can inherit all permissions in their child role node if they satisfy the parent-child relationship.

3.3.2 Common-Parent Relationship

The common-parent relationship supports the partial inheritance compared with the parent-child relationship, as shown in Fig. 2 (b).

**Definition 5** common-parent-all relationship (\( \oplus \))

If there exists a role \( r_k \) whose permission set is included in the intersection set between two roles \( r \) and \( r_j \), which is a non-empty set, then roles \( r \) and \( r_j \) are called common-parent-all roles to the role \( r_k \), where \( r \geq r_k \land r_j \geq r_k \). The relationship in such a case is denoted by \( r_k:(r \oplus r_j) \), and its semantic is shown as follows:

\[
\sigma(r_k) \subseteq (\sigma(r) \cap \sigma(r_j))
\]

As with the common-parent-all (\( \oplus^1 \)) relationship, if \( r_k:(r \oplus^1 r_j) \) and \( \forall p \in \sigma(r_k), \text{then } p \in \sigma(r) \land p \in \sigma(r_j) \). This means that senior roles \( r \) and \( r_j \) will inherit all permissions of role \( r_k \).

**Definition 6** common-parent-part relationship (\( \otimes \))

Let \( \sigma(r_{ij}) = \sigma(r_i) \cap \sigma(r_j) \) be a common permission set between roles \( r_i \) and \( r_j \). If there exists a role \( r_k \) (and 2) holds, then roles \( r_i \) and \( r_j \) are called common-parent-part roles to the role \( r_k \), denoted by \( r_k:(r_i \otimes r_j) \), where \( r \geq r_k \land r_j \geq r_k \).

\[
\left( \sigma(r_k) \cap \sigma(r_{ij}) = \sigma(r_i) \right) \land \left( \sigma(r_k) \supset \sigma(r_j) \right)
\]

\[
\land \left( \sigma(r_k) \supseteq \sigma(r_{ij}) \right)
\]

The \( r_k \) in Eq. (2) is the same as shown in Fig. 2. If \( r_k:(r_i \otimes^2 r_j) \), it means that the permission subset \( \sigma(r_k) \) in \( r_k \) are commonly inherited by roles \( r_i \) and \( r_j \). Meanwhile, there also exists a non-empty permission subset, denoted by \( \sigma(r_k) - \sigma(r_k) \), in which any permission is not owned by \( r_i \) and \( r_j \), where \( \sigma(r_{ij}) = \sigma(r_i) \cap \sigma(r_j) \cap \sigma(r_k) \). Let \( \otimes = \oplus^1 \lor \otimes^2 \) be a common-parent relationship. If there exists a role \( r_k \) and (1) or (2) holds, then \( r_i \) and \( r_j \) are common-parent roles of the role \( r_k \), denoted by \( r_k:(r_i \otimes r_j) \).

3.3.3 Common-Child Relationship

The common-child relationship is another important form of role relationships in the enterprise structure, as shown in Fig. 2 (c).

**Definition 7** common-child-all relationship (\( \oplus^1 \))

If there exists a role \( r_k \), where \( r \geq r \land r_j \geq r_j \), whose permission set is a superset of both given roles \( r \) and \( r_j \), then \( r \) and \( r_j \) are called common-child-all roles to the role \( r_k \). The relationship in such a case is denoted by \( r_k:(r \oplus^1 r_j) \), and its semantic can be defined as follows:

\[
\sigma(r_k) \supseteq (\sigma(r) \cup \sigma(r_j))
\]

**Definition 8** common-child-part relationship (\( \otimes^2 \))

Given roles \( r_i \) and \( r_j \), if there exists a role \( r_k \) and (4) holds, then \( r \) and \( r_j \) are termed common-child-part roles to the role \( r_k \), denoted by \( r_k:(r_i \otimes^2 r_j) \), where \( r_k \geq r_i \land r_k \geq r_j \).

\[
\left( \sigma(r_i) \cap \sigma(r_j) = \sigma(r_k) \right) \land \left( \sigma(r_i) \cup \sigma(r_j) = \sigma(r_k) \right)
\]

\[
\land \left( \sigma(r_k) \supseteq (\sigma(r_i) \cup \sigma(r_j)) \right)
\]

The \( r_i \) and \( r_j \) in Eq. (4) are the same as shown in Fig. 2. Let \( \otimes = \oplus^1 \lor \otimes^2 \) denote a common-child relationship. If there exists a role \( r_k \) and (3) or (4) holds, then roles \( r_i \) and \( r_j \) are common-child of the role \( r_k \), denoted by \( r_k:(r_i \oplus r_j) \).
3.3.4 Transformation of Role Relationships

From definitions of role relationships above, it can be observed that there are certain relations among them. As shown in Fig. 3, if \( r_k \vdash r_i \otimes r_j \), both roles \( r_i \) and \( r_j \) can be viewed as a parent node of the role \( r_k \) since they can possess all of the permissions assigned to \( r_k \). Similarly, if \( r_k \vdash r_i \otimes r_j \) and \( r_k \)'s permission set is divided into two subsets \( \sigma(r_k) \) and \( \sigma(r_k \vdash r_i) \), then the permission set associated with \( r_k \) can be common inherited by roles \( r_i \) and \( r_j \), while the permission set of associated with the role \( r_k \) cannot be inherited by them, in which \( \sigma(r_k \vdash r_i) = \sigma(r_k) \cap \sigma(r_i) \) and \( \sigma(r_k \vdash r_j) = \sigma(r_k) \cap \sigma(r_j) \). The following lemma captures the relationship between common-parent and parent-child operator, \( \rightarrow \) and \( \otimes \), respectively.

**Lemma 1:** If \( r_k \vdash r_i \otimes r_j \), then \( r_k \rightarrow r_i \) and \( r_k \rightarrow r_j \); if \( r_k \vdash r_i \otimes r_j \), then \( r_k \rightarrow r_i \) and \( r_k \rightarrow r_j \), where \( \sigma(r_k) = \sigma(r_k) \cap (\sigma(r_i) \cup \sigma(r_j)) \).

Similarly, if \( r_k \vdash r_i \otimes r_j \), then \( r_k \rightarrow r_i \) and \( r_k \rightarrow r_j \); if \( r_k \vdash r_i \otimes r_j \), then \( r_k \rightarrow r_i \) and \( r_k \rightarrow r_j \), where \( \sigma(r_k) = \sigma(r_k) \cap (\sigma(r_i) \cup \sigma(r_j)) \).

3.4 The Concept of Atom Role

Something brings difficulties in the transformation of role relationships. The reasons for that are each role may have three permission types, and a role often have different role relationships with different roles. A typical example is the regular role \( r_k \) in Fig. 1: there is a relationship \( r_k \vdash (r_i \otimes r_j) \), so a division operation on \( r_k \) is needed by the transformation from operator \( \otimes \) to \( \rightarrow \), as for the relationship \( r_k \vdash (r_i \otimes r_j) \), \( r_k \) also requires to be divided in order to make this transformation from operator \( \otimes \) to \( \rightarrow \). It can be considered that multiple dividing-operations for a regular role may happen in the whole process of role’s relationship transformation. However, the division of regular role is closely related to permission inheritance among roles. Hence, we can’t divide the role only by the set of permissions of original role. To organize and implement transformation of role relationships effectively, we present the notion of permission inheritance path which reflects the inheritance relationship of role’s permissions in the whole hierarchy structure of role.

**Definition 9** permission inheritance path

A permission inheritance path for given permission \( p \) of regular role \( r_i \), is of the form \( \sigma(r_i) \vdash p \rightarrow \sigma(r_{i+1}) \vdash p \rightarrow \sigma(r_{i+2}) \vdash p \rightarrow \cdots \rightarrow \sigma(r_{i+n}) \vdash p \rightarrow \sigma(r_j) \vdash p \), where \( 0 \leq n \), and \( r_j \) refers to a top senior role of \( r_i \), \( \sigma(r_i) \vdash p \rightarrow \sigma(r_{i+1}) \vdash p \) means that the permission \( p \) of \( r_i \) can be inherited by \( r_{i+1} \) via the parent-child (\( \rightarrow \)) relationship.

Now that a permission \( p \) can be common inherited by multiple senior roles, \( p \vdash \sigma(r_i) \) may have several permission inheritance paths. Let \( P\text{path}(r_i, p) \) denote the permission inheritance path set for \( r_i \)'s permission \( p \). In this paper, an attribute, called Path Role set (\( PR\text{set} \)), which refers to the set of all roles that compose the permission inheritance path, is affiliated to each \( P\text{path}(p) \), e.g., with an inheritance path such that \( \sigma(r_i) \vdash p \rightarrow \sigma(r_{i+1}) \vdash p \rightarrow \sigma(r_{i+2}) \vdash p \rightarrow \cdots \rightarrow \sigma(r_{i+n}) \vdash p \rightarrow \sigma(r_j) \vdash p \), it can be seen that \( P\text{path}(r_i, p) \vdash PR\text{set} = \{ r_1, r_4, r_7, r_6 \} \). Suppose \( path_k(r_i, p) \in P\text{path}(r_i, p) \) and \( path_l(r_j, p) \in P\text{path}(r_j, p) \). If their attribute sets, \( path_k(r_i, p) \vdash PR\text{set} \) and \( path_l(r_j, p) \vdash PR\text{set} \) respectively, have the same value of each component, it is said that this two \( PR\text{set} \)s associated with permission inheritance paths are equal, written as \( path_k(r_i, p) \vdash PR\text{set} = path_l(r_j, p) \vdash PR\text{set} \).

**Definition 10** inheritance path relationships \((\approx, \leq)\)

Let \( \sigma = \{ path_k(r_i, p) \vdash path_l(r_j, p) \in P\text{path}(r_i, p) \}, 1 \leq k \leq n \} \) and \( \omega = \{ path_k(r_i, p) \vdash path_l(r_j, p) \in P\text{path}(r_i, p) \}, 1 \leq l \leq m \} \) represent \( P\text{path}(r_i, p) \vdash \vdash PR\text{set} \) set and \( P\text{path}(r_i, p) \vdash \vdash PR\text{set} \) set, respectively. Where \( n, m \) represent the number of inheritance paths in \( P\text{path}(r_i, p) \vdash \vdash PR\text{set} \) and \( P\text{path}(r_i, p) \vdash \vdash PR\text{set} \) respectively. A similar-inheritance path relationship exists between \( p_i \) and \( p_j \) of regular role \( r \), denoted by \( p_i \approx p_j \), if and only if every corresponding \( PR\text{set} \) set of each permission inheritance path set is a member of the other. If not, \( p_i \) and \( p_j \) satisfy a dissimilar-inheritance path relationship, written as \( p_i \neq p_j \). The semantic of the relationship \( "\approx" \) is defined as follows:

\[
 r_i \vdash r_j \iff \forall 0 \leq k \leq m, \exists 0 \leq l \leq n \exists \text{path}(r_i, p) \vdash \vdash \text{PRset} \wedge (|\sigma| = |\omega|)
\]

From the definition above, if \( r_i \approx r_j \) and \( r_i \approx r_p \), then \( r_j \approx r_p \). In other words, the relationship \( "\approx" \) has transitive property.

**Theorem 1:** Let \( \sigma(r) \) be the set of all permissions in regular role \( r \), then the following conclusions are drawn:

(i) \( \forall p, p_i \in \sigma(r), PB(p)_{p_i} \approx p_j \iff \forall p, p_i \in \sigma(r), PR(pp)p_k \approx p_l \)

(ii) \( \forall p, p_i \in \sigma(r), PR(p)_{p_i} \approx PB(p)p_k \approx p_l \)
The meaning of the above four conclusions will be given as below:

- Conclusion (i) means that there exists a similar-inheritance path relationship between any two public permissions (or private permissions) \( p_i, p_j \) of a regular role \( r \).
- Conclusion (ii) means that there exists a dissimilar-inheritance path relationship between any private permission \( p_i \) and public permission \( p_k \) of a regular role \( r \).
- Conclusion (iii) means that given any selective permission \( p_i \) and public permission \( p_k \) of a regular role \( r \), if \( p_i \) can be inherited by any role \( x \) in any inheritance path of \( p_k \), then there exists a similar-inheritance path relationship between \( p_i \) and \( p_k \).
- Conclusion (iv) means that given any selective permission \( p_i \) and public permission \( p_k \) of a regular role \( r \), if there is a role \( x \) that cannot inherit \( p_i \) in an inheritance path of \( p_k \), then there exists a dissimilar-inheritance path relationship between \( p_i \) and \( p_k \).

**Proof.** By characteristics of public and private permissions respectively have a unified set of permission inheritance paths, then clearly conclusion (i) and (ii) are tenable.

It is possible that public permission \( p_l \) has several permission inheritance paths, denoted by \( path \! (p_k) \), where \( 1 \leq l \leq m \). In accordance with Definition 10, if any selective permission \( p_l \) needs to meet the relationship \( \sim \) with \( p_k \), then \( p_l \) must be inherited by roles in every \( PRset \) of \( path \! (p_k) \), therefore, conclusion (iii) is true. On the contrary, as long as there exists one role that cannot inherit \( p_l \) in any one of \( PRset \) set associated with \( path \! (p_k) \), which obviously contradict with Definition 10. Therefore, conclusion (iv) is also true.

The inheritance path relationships specify whether different permissions in a regular role can be inherited by senior roles along with the same permission inheritance path. The relationship \( \sim \) among permissions supports this commonality, while the relationship \( \propto \) does not. \( \forall p_i, p_j \in \sigma(r) \), the inheritance way of regular role \( r \) is said to be atomic if \( r.p_i \approx r.p_j \). From Theorem 1, it can be observed that the inheritance for each regular role is not usually atomic in a complex role structure. For instance, given a public permission \( p_i \) and a private permission \( p_j, p_i \) meet the relationship \( \propto \) with \( p_j \).

**Definition 11 atom role**

A role is formed by dividing the regular role, which only contains part of permissions assigned to the regular role. A regular role \( r \) can be divided into one or more atom roles according to inheritance path relationships among its permissions, denoted by \( AR \sim (r) \), where \( i \leq f \leq i + n \). Moreover, all permissions of an atom role satisfy the relationship \( \sim \), while any two permissions in different atom roles, which are gotten by dividing the same regular role, satisfy the relationship \( \propto \). The definition of the atom role is represented as follows:

\[
(i) \forall p_l \in \sigma(\text{AR}) \left( \sigma(\text{AR}i) \cup \sigma(\text{AR}j) \right)
\]

\[
(ii) \forall p_l \in \sigma(\text{p}(r)), p_j \in \sigma(\text{p}(r)) \left( p_l \approx p_j \right)
\]

\[
(iii) \forall p_l \left( p_l \in \sigma(\text{p}(r)), p_j \in \sigma(\text{p}(r)) \right) \left( p_l \propto p_j \right)
\]

\[
(iv) \forall p_l \left( p_l \in \sigma(\text{p}(r)), p_j \in \sigma(\text{p}(r)) \right) \left( p_l \propto p_j \right)
\]

\[
\text{TPSP}(r, p_i, x) \implies (p_i \approx p_k)
\]

\[
\text{FPSP}(r, p_i, x) \implies (p_i \propto p_k)
\]

Figure 4 shows corresponding atom roles for regular roles \( r_1 \) and \( r_2 \) based on their inheritance path relationships in Fig. 1, e.g., suppose the permission sets as described in Definition 10.

\[
\text{psource}(ar: \text{ARS}) \rightarrow \sigma(ar) \text{ denotes a function mapping from an atom role to its permission source.}
\]

**Definition 13 atom role relationships \( (\geq) \)**

\[
(r, \text{AR}) = (r', \text{AR}) \iff \text{psource}(r, \text{AR}) = \text{psource}(r', \text{AR})
\]

\[
(r, \text{AR}) \geq (r', \text{AR}) \iff \begin{cases} \text{psource}(r, \text{AR}) \geq \text{psource}(r', \text{AR}), r \geq r' \\ \forall p_l \in \sigma(\text{AR}r) \left( p_l \in \text{PBP}(r') \cup \text{TPSP}(r', p, r), r \geq r' \right) \end{cases}
\]

**Theorem 2:** The relationship \( (\geq) \) over atom role set \( AR \) is a partial order.

**Proof.**

(i) Reflexivity:

Form the Definition 13, observe that \( (r, \text{AR}) \geq (r, \text{AR}) \) for all \( (r, \text{AR}) \in AR \).

(ii) Antisymmetry:

Suppose \( (r, \text{AR}) \geq (r', \text{AR}) \) and \( (r', \text{AR}) \geq (r, \text{AR}) \).

Then

\[
(r, \text{AR}) \geq (r', \text{AR}) \iff \begin{cases} \text{psource}(r, \text{AR}) \geq \text{psource}(r', \text{AR}), r \geq r' \\ \forall p_l \in \sigma(\text{AR}r) \left( p_l \in \text{PBP}(r') \cup \text{TPSP}(r', p, r), r \geq r' \right) \end{cases}
\]

\[
(r', \text{AR}) \geq (r, \text{AR}) \iff \begin{cases} \text{psource}(r', \text{AR}) \geq \text{psource}(r, \text{AR}), r' \geq r \\ \forall p_l \in \sigma(\text{AR}r') \left( p_l \in \text{PBP}(r) \cup \text{TPSP}(r, p, r'), r' \geq r \right) \end{cases}
\]
If (8) and (10) hold, then \((r, \text{AR}_i) = (r', \text{AR}_i)\).

If (9) and (11) hold, then \(r = r'\). From the definition of relationship \(\geq\), it can be seen that all permissions in \(\sigma(\text{AR}_i)\) and \(\sigma(\text{AR}_j)\) are public permissions of the regular role \(r\), then clearly \((r, \text{AR}_i) = (r', \text{AR}_j)\).

Similarly, (8) and (11) or (9) and (10) can hold together if and only if \((r, \text{AR}_i) = (r', \text{AR}_j)\). This follows from the observation that

\[
(r \geq r') \land (r' \geq r) \Rightarrow (r = r')
\]

\[
\Rightarrow (p_{\text{source}}(r, \text{AR}_i) \cap p_{\text{source}}(r', \text{AR}_j) = \emptyset)
\]

and \(\forall p \in \text{PBP}(r') \lor \forall p \in \text{PBP}(r')\).

\[
\Rightarrow p_{\text{source}}(r, \text{AR}_i) = p_{\text{source}}(r', \text{AR}_j)
\]

Therefore, the relationship \(\geq\) is antisymmetrical.

(iii) Transitivity:

Note that if \((r, \text{AR}_i) \geq (r', \text{AR}_j)\) and \((r', \text{AR}_j) \geq (r'', \text{AR}_k)\), then

\[
(r', \text{AR}_j) \geq (r'', \text{AR}_k) \Leftrightarrow
\begin{align*}
( & p_{\text{source}}(r', \text{AR}_j) \supseteq p_{\text{source}}(r'', \text{AR}_k), r' \geq r'' \quad (12) \\
& \forall p \in \text{PBP}(r') \lor \forall p \in \text{PBP}(r'') \lor \forall p \in \text{TPS}(r'', p, r'), r' \geq r'' \quad (13)
\end{align*}
\]

If (8) and (12) hold together, we can easily conclude that \(r \supseteq r' \supseteq r''\) and \(p_{\text{source}}(r, \text{AR}_i) \supseteq p_{\text{source}}(r'', \text{AR}_k)\), which implies \((r, \text{AR}_i) \geq (r'', \text{AR}_k)\).

All permissions of \(\text{AR}_k\) are either public permissions or selective permissions that can be inherited by \(r'\), and each permission of \(\text{AR}_j\) is either public permission of \(r'\) or selective permission that can be inherited by \(r\), if (9) and (13) hold together. By properties of public and selective permissions, we can conclude that \((r, \text{AR}_i) \geq (r'', \text{AR}_k)\).

If (8) and (13) hold together, and if each permission of \(\text{AR}_k\) is a public permission of \(r'\), then it must be true that \((r, \text{AR}_i) \geq (r'', \text{AR}_k)\) since \(r \supseteq r' \geq r''\) and \(\forall p \in \text{PBP}(r'') \Rightarrow p_{\text{source}}(r, \text{AR}_i) \supseteq p_{\text{source}}(r'', \text{AR}_k)\). Otherwise, suppose that there exists a selective permission \(p\) such that \(p \in p_{\text{source}}(r', \text{AR}_j) \land p \notin p_{\text{source}}(r, \text{AR}_i)\) in \(\sigma(\text{AR}_i)\). But then \(p_{\text{source}}(r, \text{AR}_i) \supseteq p_{\text{source}}(r', \text{AR}_j)\) defined by \((r, \text{AR}_i) \geq (r', \text{AR}_j)\), a contradiction. Thus, we have \((r, \text{AR}_i) \geq (r'', \text{AR}_k)\) since \(\forall p \in \text{PBP}(\text{AR}_k) \subseteq p_{\text{source}}(r, \text{AR}_i)\). Similarly, if (9) and (12) hold that \((r, \text{AR}_i) \geq (r'', \text{AR}_k)\).

In a word, the relationship \(\geq\) over \(\text{ARS}\) is reflexive, antisymmetric, and transitive, therefore it is a partial order.

\textbf{Definition 14 Seniors(\text{AR}_i)}

The set of Senior atom roles for a given atom role \(\text{AR}_i\) is all \(\text{AR}_j\) such that \(\text{AR}_j \supseteq \text{AR}_i\), denoted by \text{Seniors}(\text{AR}_i).

\textbf{Definition 15 Juniors(\text{AR}_i)}

The set of Junior atom roles for a given atom role \(\text{AR}_i\) is all \(\text{AR}_j\) such that \(\text{AR}_j \supseteq \text{AR}_i\), denoted by \text{Juniors}(\text{AR}_i).

By Theorem 2, it has proved that the atom role set \(\text{ARS}\) together with the relationship \(\geq\) forms a hierarchy structure, called AtomRole Hierarchy Structure in this paper, denoted by \(\text{ARH} = (\text{ARS}, \geq)\). For the role structure in Fig. 1, Fig. 5 shows its corresponding atom role hierarchy structure.

\textbf{3.5 Generation Algorithm}

Before atom roles are generated, all permissions \(\sigma(r)\) that each role possesses including permissions inherited from junior roles should be generated on the basis of permission inheritance paths. Dividing regular roles into atom roles based on permission inheritance paths, we can guarantee that the relationship \(\geq\) over atom role set is a partial order according to Theorem 2; and that the atom role generated contains the largest permissions that can be inherited by senior roles in the meantime, which can avoid generating overfull atom roles. In this section, we describe a generation algorithm to divide regular roles into atom roles dynamically based on their permission inheritance paths. The notation \text{PARS} refers to a set of current atom role.

From Definition 10, it follows that \(p\) will be a member of the permission set \(\sigma(r)\) if a permission inheritance path passing through regular role \(r\) for a given permission \(p\). Moreover, given a permission set \(\sigma(r_i)\), if multiple regular roles \(r_j\), where \(i \leq j \leq i + n\), contain \(\sigma(r_i)\), then all permissions of \(\sigma(r_i)\) satisfy the relationship \(\geq\) since they share the following permission inheritance path: \(\sigma(r_i) \supseteq \sigma(r_{i+1}) \supseteq \sigma(r_{i+2}) \supseteq \ldots \supseteq \sigma(r_{i+n})\), \(p\), that is, \(r_i\) is a current atom role associated with \(r_j\). On the other hand, let \(\sigma(r_j)\) denote the complement set of \(\sigma(r_i)\) in \(\sigma(r_j)\), any permission in \(\sigma(r_j)\) only belongs to corresponding regular roles’ permission sets, e.g., for each permission \(p\) in \(\sigma(r_j)\) is only a member of \(\sigma(r_i)\); therefore, any two permissions in each complement set apparently satisfy the relationship \(\geq\). This means that \(r_j\) is also a current atom role associated with \(r_j\).

From the discussion above, the basic idea of the generation algorithm for atom roles is as follows: select an element \(r\) from the \text{Roles} set one by one, if there exists a non-empty common permission subset between \(r\) and \(\text{AR}_i\), which is a member of the \text{PARS} set, for \(i = 1, 2, \ldots\), then recalculate current atom roles by performing the following two operations: calculating intersection of \(\sigma(r)\) and \(\sigma(\text{AR}_i)\), and calculating complement of \(\sigma(\text{AR}_i)\) in \(\sigma(r)\), and then a new \text{PARS} set is generated. Otherwise, if there is no common permission between \(r\) and any element of \text{PARS}, then
rp is directly viewed as a new current atom role and added to the \textit{PARS}. When all elements of \textit{Roles} are traversed, the algorithm finished.

The notations \textit{sflag} and \textit{pflag} are used to denote regular role’s scan-mark and atom role’s operation-mark, respectively. If \textit{r.sflag} = \textit{true}, it indicates that the regular role \textit{r} has already been selected from \textit{Roles}. And if \textit{AR.pflag(r)} = \textit{true}, it means, two operations, calculating complement of \textit{\sigma(AR)} in \textit{\sigma(r)} and calculating intersection of \textit{\sigma(r)} and \textit{\sigma(AR)}, have been made. Let edge\textit{(r, AR)} be a direct edge from \textit{r} to \textit{AR}, where \textit{AR} is a current atom role associated with \textit{r}, and edge\textit{(AR, AR)} is a direct edge from \textit{AR} to \textit{AR}, where \textit{AR} is obtained by redividing-operation with respect to \textit{AR}. The generation algorithm of atom roles is as follows:

\textbf{step1:} Initialize \textit{PARS} to be an empty set, and for each \textit{r} \in \textit{Roles}, \textit{r.sflag} = \textit{false}.

\textbf{step2:} Choose a regular role from \textit{Roles}, denoted by \textit{rbegin}, where \textit{rbegin.sflag} = \textit{false}. Then let \textit{\sigma(ARbeg)} = \textit{\sigma(rbegin)}, and \textit{PARS} = \textit{PARS} \cup \{\textit{ARbeg}\}, \textit{rbegin.sflag} = \textit{true}. Add the edge: edge\textit{(rbegin, ARbeg)}.

\textbf{step3:} Choose any existing \textit{ri} (from \textit{Roles}) with clause \textit{ri.sflag} = \textit{false}. If \textit{ri.sflag} = \textit{true} for each \textit{rj} \in \textit{Roles}, then turn to step8.

\textbf{step4:} Let \textit{r.sflag} = \textit{true}. If \textit{AR.sflag(r)} = \textit{true} for each \textit{AR} \in \textit{PARS}, then turn to step7, else find a current atom role from \textit{PARS}, written as \textit{AR}, where \textit{AR} is obtained by redividing-operation with respect to \textit{AR}. The generation algorithm of atom roles is as follows:

\textbf{step5:} Let \textit{AR.sflag} = \textit{true}, if \textit{\sigma(AR)} is empty, then turn to step4.

\textbf{step6:} Let \textit{\sigma(r)} = \textit{\sigma(r)} - \textit{\sigma(AR)}, \textit{\sigma(AR')} = \textit{\sigma(AR)} - \textit{\sigma(AR)}, and \textit{PARS} = \textit{PARS} \cup \{\textit{AR, AR'}\} - \{\textit{AR}\}. Then add the following three edges: edge\textit{(r, AR)}, edge\textit{(AR, AR')}, edge\textit{(AR, AR')}.

\textbf{step7:} If \textit{\sigma(r)} is not empty, then let \textit{\sigma(AR)} = \textit{\sigma(r)} and \textit{PARS} = \textit{PARS} \cup \{\textit{AR}\}. Following that, add the edge: edge\textit{(r, AR)}, \forall \textit{VAR} \in \textit{PARS}, let \textit{AR.pflag} = \textit{false} and turn to step3.

\textbf{step8:} Clear up the marks of all nodes. After that, the algorithm is finished.

In view of the role structure in Fig. 1, Fig. 6 describes the generation process of atom roles, where \textit{rbegin} = \textit{r1}.

3.6 Model Structure

The structure of ATRBAC model is shown in Fig. 7, in which \textit{UAR}, \textit{RAR}, \textit{UAA}, \textit{RAS}, \textit{ARP}, \textit{PAR} represent the relations between users and regular roles, regular roles and atom roles, users and atom roles, regular roles and permissions, atom roles and permissions, respectively. \textit{ARH} refers to the atom role hierarchy. In ATRBAC model, when a role is assigned to a certain user through \textit{UAR}, the user’s permissions authorized, compared with traditional models, will be determined by \textit{RAR}, \textit{ARP} and \textit{ARH} out of the security consideration for not violating the least privilege principle, rather than role hierarchy (\textit{RH}) and permission assignment (\textit{PA}). This is because in large-scale distributed systems, the structures of regular roles are no longer come into being hierarchical organizations because of complex role relationships and multi-level requirements in applications.

Atom roles, which only contain partial permissions of regular roles, are obtained from dividing regular roles according to inheritance path relationships among their permissions. In addition, the atom role hierarchy structure is built on the basis of diversity of permission types and role relationships, consisting of the atom roles set \textit{ARS} together with relationship “\geq”, so it can suit the enterprises and organizations in complex situations. In ATRBAC, the relationship “\geq” can be seen as an atom-role-atom-role authorization in which the parent atom role is automatically authorized to the permissions of the child atom role, so there is no need to assign the same permission to parent atom role after assigning that to the child atom role. This could reduce the expenses caused by repeated authorization and enhance the flexibility of access control in practical administration.
4. Model Analysis

Take the specific application in Fig. 1 as an example, we compare ATRBAC model with other three models: RBAC2 [4], EHRBAC [12] and RPI-RBAC [13] in the following properties [9], as shown in Table 2. The notation ‘O’ means that the model can easily achieve the property. The notation ‘Δ’ means that the model need a detailed and cautious configuration to support the property. In other words, if design of roles is not perfect, the property is hardly achieved.

(i) Fine-grained authorization: The management and control for user permissions is at the role-level in RBAC2, thus it is a coarse-grained authorization model that lacks flexibility and efficiency. For purposes of supporting incomplete inheritance and fine-grained authorization, ATRBAC could divide the regular role into one or more atom roles based upon inheritance path relationships among its permissions, and the atom role contains part of permissions associated with the regular role. This means that a user authorized to a regular role \( r \) may only perform partial permissions subset of \( r \), depending on relationships of \( RAR \), \( ARP \) and \( ARH \). For example, in Fig. 1, assume that Scott is a user of \( r_7 \), we have that Scott’s permission set \( \sigma(r_7) = \{p_1, p_5, p_{11}, p_{12}\} \), instead of \( \sigma(r_7) = \{p_1, p_5, p_6, p_{11}, p_{12}\} \), since the permission \( p_6 \) is a private permission of \( r_4 \) that can’t be inherited by the senior role \( r_7 \).

(ii) Hierarchy administration: RBAC2 model, built on the unconditional inheritance between regular roles, is unable to describe the complex role structure effectively. Similarly, in RPI-RBAC, the model also does not have the hierarchy administration features, because properties of permissions have not been taken into consideration in the process of role division and the spread of permissions can only be administered from both horizontal and vertical directions. Unlike them, in ATRBAC, atom roles have a complete hierarchical structure, which means that after assigning the permissions to the child atom role, the same permission will not be assigned to the parent atom role, thus the expenses in the repeated authorization could be avoided and the flexibility of access control could be enhanced. For example, in the atom role hierarchy with respect to the role structure in Fig. 1 (see Fig. 5), the atom role \( AR_{13} \) will inherit all permissions of the atom role \( AR_{11} \) since \( AR_{13} \geq AR_{11} \).

(iii) The least privilege and separation of duties principle: In EHRBAC, now that the senior role is allowed to inherit all permissions of the junior role by the extended inheritance, and if a user is authorized to a supervisor role \( r \), he may accumulate all the permissions, which were directly assigned to \( r \) or indirectly inherited from its junior roles. Hence, like RBAC2, it could lead to the problem of permission abuse. To the contrary, in ATRBAC model, the atom role will only inherit the permissions of its junior atom roles. Therefore, the atom role could satisfy the need of having its own unique permission and avoid the abuse of permissions. For example, suppose Jim is a member of \( r_3 \) and \( p_4 \) is a private permission out of the need for privacy (see Fig. 1). By Theorem 1, we have \( p_1 \not\propto p_4 \), so \( r_3 \) should be divided into two atom roles. Let \( \sigma(AR) = \{p_4\} \), it can be seen that \( Seniors(AR_r) = \emptyset \), according to \( RAR \), \( ARP \) and \( ARH \) in ATRBAC, where \( \emptyset \) is an empty set. This indicates that all permissions of \( AR_r \) cannot be inherited by senior regular roles. Hence, Jim has his own permission \( p_4 \). By dividing the permission set of each regular role, the conflict permissions can be divided into different atom roles; atom roles that have conflict permissions are called mutually exclusive atom roles which can’t be assigned to the same user. In ATRBAC model, separation of duties is achieved on the basis of mutually exclusive atom roles, which can be realized by adding restraint condition to the assignment of atom roles. For example, in Fig. 5, assume that \( AR_7 \) and \( AR_{10} \) is a pair of mutually exclusive atom roles and John is a member of \( AR_7 \). If John further requires for the assignment of \( AR_{10} \), then there be a conflict between \( AR_7 \) and \( AR_{10} \), thus this authorization request will be refused.

(iv) Scalability: Atom role is formed by dividing the regular role. In permission administration systems, note that if the administration strategy allows the senior role to unconditionally inherit permissions of the junior role, it follows that all permissions in a regular role have same permission inheritance paths, so it is a one-to-one correspondence between regular roles and atom roles. Just as extended inheritance can be degenerated into normal inheritance, RBAC2 could be regarded as a special case of ATRBAC.

### Table 2

<table>
<thead>
<tr>
<th>Properties</th>
<th>ATRBAC</th>
<th>RBAC2</th>
<th>EHRBAC</th>
<th>RPI-RBAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy Administration</td>
<td>O</td>
<td>Δ</td>
<td>O</td>
<td>Δ</td>
</tr>
<tr>
<td>Least Privilege</td>
<td>O</td>
<td>Δ</td>
<td>Δ</td>
<td>O</td>
</tr>
<tr>
<td>Separation of Duties</td>
<td>O</td>
<td>Δ</td>
<td>Δ</td>
<td>O</td>
</tr>
<tr>
<td>Fine-Grained</td>
<td>O</td>
<td>Δ</td>
<td>O</td>
<td>Δ</td>
</tr>
<tr>
<td>Scalability</td>
<td>O</td>
<td>Δ</td>
<td>O</td>
<td>Δ</td>
</tr>
</tbody>
</table>

5. Conclusion

Since the traditional role-based hierarchy model is unable to effectively reflect the complex role structure, and the role inheritance with unlimited features violates the least privilege principle, this paper, based on RBAC model, studied role relationships and their transformation method, proposed the concept of atom role by dividing regular roles according to inheritance path relationships among their permissions, and finally constructed a new atom-role-based access control model (ATRBAC). Compared with other models, we can see that, the model not only has an atom role hierarchi-
cal structure, which enhances the flexibility in access control, but also effectively avoids the abuse of permissions. Therefore, ATRBAC model is able to fully exert the various functions of RBAC and well meet the needs of practical applications.

In large-scale distributed systems, the delegation authority mechanism is an effective solution to optimize the administration authorization, therefore, an in-depth study on the delegation relationship and constraint analysis on atom roles will be our next focus. In this paper, we intend to give an atom roles’ generation algorithm that could be realized simply and easily, and an in-depth study on more effective and intuitive generation algorithm and update algorithm will also be our next focus.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant 61103244), in part by the Guangdong Natural Science Foundation (Grant S2011010004197), and in part by the Cooperation Project in Industry, Education and Research of Guangdong Province and Ministry of Education of China (Grant 2009B090300345).

References


Weihong Cai was born in Chaozhou Guangdong province of China in 1963. He is a professor of Shantou University. His research interests include network and communications, information security.

Richeng Huang was born in Zhanjiang Guangdong province of China in 1986. He is a Master candidate of Shantou University. His research interests include network and communications, information security.

Xiaoli Hou was born in Anyang Henan province of China in 1986. She is a Master candidate of Shantou University. Her research interests include network and communications, information security.

Gang Wei was born in Binyang Guangxi province of China in 1963. He is a doctoral tutor of South China University of Technology. His main interests include communications, information processing theory and technology.
Shui Xiao was born in Zhuzhou Hunan province of China in 1983. He is a Master candidate of Shantou University. His research interests include network and communications.

Yindong Chen was born in Jieyang Guangdong province of China in 1983. He is an associate professor in Shantou University. His research interests include communications, cryptography, and information security.