 CMC, vol.60, no.1, pp.395-408, 2019

A Cross-Tenant RBAC Model for Collaborative Cloud Services

Zhengtao Liu[1, *] and Jinyue Xia[2]

Abstract: Tenants in the cloud computing environment share various services, including storage, network, computing, and applications. For better use of services in the cloud computing environment, tenants collaborate in tasks, resulting in challenges to the traditional access control. This study proposes a cross-tenant role-based access control (CT-RBAC) model for collaborative cloud services. This model covers the CT-RBAC0, CT-RBAC1, CT-RBAC2, and CT-RBAC3 models. The model not only extends the RBAC model in the multi-tenant cloud computing mode but also includes four types of authorization modes among tenants. Consequently, the role inheritance constraint is increased, and fine-grained authorization access among trusted tenants is realized.

Keywords: Cloud computing, multi-tenant, collaboration, fine-grained authorization.

1 Introduction

Cloud computing [Nist (2015); Wood, Ramakrishnan and Shenoy (2015)] is a mode that pays for usage. This mode provides useful, convenient, and on-demand network accesses into configurable computing resource-sharing pools (resources include network, server, storage, application software, and services). These resources can be offered quickly and require a few management works or interactions with service suppliers. With unique advantages, such as broadband internet, resource pool sharing, flexible configuration, service on demand, and pay for service, cloud computing can lower the maintenance costs of users for computing and storage and reduce the constraints brought by restricted storage and computing resources of users.

In the cloud computing environment, clients (tenants) deploy all services or data on the cloud computing platform of the cloud computing supplier, such that tenants share various services, including storage, network, computing and applications, thereby reducing the cost and increasing efficiency. Tenants are isolated mutually, presenting a challenge to their collaboration. Hence, the corresponding safety mechanism and system structure must be designed to protect the confidentiality, integrity, and usability of user data [Curry, Darbyshire, Fisher et al. (2010); Ghazizadeh, Zamani, Manan et al. (2013)].

Access control protects information resources from illegal use and access by restricting the information access capability and scope of users. The access control technology under

[1] School of Computer Science and Engineering, Sanjiang University, Nanjing, 210012, China.

[2] International Business Machines Corporation, New York, USA.

[*] Corresponding Author: Zhengtao Liu. Email: lzt_jh_cn@163.com.

the traditional computing mode can effectively protect information resources from illegal access. Research on access control under the cloud computing environment has developed with cloud computing. Before cloud computing, numerous scholars have studied the access control problem in grids. Since grid technology was replaced by cloud computing technology, the research has shifted to the access control of cloud computing. Cloud computing features numerous safety problems. Thus, access control is the core content of cloud computing security.

2 State-of-the-art

Given that tenants share physical resources and the difficulty in achieving reliability of resources, tenants can gain useful information from the bottom physical resources through side channel attacks [Meghanathan (2013)]. In addition, the deployment of access control strategies on a virtual machine might cause conflicts among multiple tenants accessing resources, thus resulting in unapproved information flow or information flow with incorrect authorization distribution on the physical host. In the cloud environment, communication among different tenants shall be guaranteed by access control. Moreover, each tenant possesses a unique access control strategy, which increases the complexity of access control on the entire cloud platform.

Different tenants can access different applications and computing resources on the same cloud server. Access control technology can be regressed to studies on the model to avoid access control failure. Many scholars have added multi-tenant technology into the traditional access control model to generate new access control models and strengthen the data access control of tenants. Yang et al. [Yang, Lai and Lin (2013)] proposed the multi-tenant role-based access control (RBAC) model that determines the identity and role of users through identity management and realizes the independence of application programs and data isolation through the access rights of tenants. Users in the cloud environment are divided into ordinary users and resource owners according to the structure of cloud computing. Users could only access corresponding resources upon authorization of resource owners [Tang, Wei, Sallam et al. (2012)]. Li et al. [Li, Shi and Guo (2010)] suggested separating the security duty between the cloud service supplier (CSP) and the tenant (client) and proposed an access control model based on multi-tenants. In this model, CSP can add, delete and manage tenants in the cloud and process relevant security problems. The model manages the access control of tenants and assures the security of tenants through access control. Wang et al. [Wang, Wang, Guo et al. (2018)] proposes a data access control model for individual users. Through the semantic dependency between data and the integration process from bottom to top, the global visual range of inverted XML structure is realized.

The above model realizes access control of multi-tenants but overlooks the collaboration of multiple tenants. The access control problems in collaboration have been extensively addressed in traditional environments. Various extensions of RBAC have been proposed to enable multi-domain access control [Cohen, Thomas, Winsborough, et al. (2002); Li, Zhang, Xu et al. (2009); Lin, Rao, Bertino et al. (2008); Zhang, Zhang and Sandhu (2006)]. These methods must define the collaboration strategies of different domains by centralized means. However, users of cloud computing originate from different

organizations and utilize independent management authorization platforms. Therefore, these methods cannot adequately adapt to the characteristics of cloud computing.

Current Infrastructure as a Service (IaaS) cloud platforms have their own authorization system, containing different access control policies and models. Clients with accounts in multiple cloud providers struggle to manage their rules in order to provide a homogeneous access control experience to users. Sette et al. [Sette, Chadwick and Ferraz (2017)] proposes a solution: an Authorization Policy Federation (APF) of heterogeneous cloud accounts. To realize collaboration among different tenants, Tang et al. [Tang (2013)] proposed the administration of a multi-tenant approval system (MTAS) in combination with the RBAC model. Trust conditions were added to the AMTAS model, which performs formal analysis of trust among different tenants, based on MTAS. Later, the MT-RBAC model was proposed based on AMTAS [Tang, Sandhu and Li (2015)]. This model extends the traditional RBAC model, increases two built-in components (issuer and tenant), and realizes collaboration of different tenants by setting up the trust relationship among different tenants. The MT-RBAC model integrates three different tenant trust models, namely, MT-RBAC0, MT-RBAC1, and MT-RBAC2. MT-RBAC0 is the basic model and requires all trustors to gather all roles and expose corresponding authorization to the trustee. To restrict unnecessary authorization information of trustors, the MT-RBAC1 model divides the role set into two subsets, in which the public role is exposed to all trustees. The MT-RBAC2 model provides more detailed constraints and offers different role authorization sets according to the trust level of trustees.

Based on the MT-RBAC, a cross-tenant RBAC (CT-RBAC) model is proposed in this study. Based on the extensively used RBAC96 [Sandhu (1997)] (RBAC0, RBAC1, RBAC2, and RBAC3), the proposed CT-RBAC model is used to design the CT-RBAC0, CT-RBAC1, CT-RBAC2, and CT-RBAC3 models. Compared with the MT-RBAC model, the CT-RBAC model not only considers different types of authorization modes among different tenants, the exposure of users and role information in authorization, and management of role inheritance but also extends the RBAC model in the multi-tenant cloud computing mode.

3 Equations and mathematical expressions

The CT-RBAC model covers the unilateral trust relationships among different types of tenants. Trustors and trustees can set up flexible trust relationships according to practical demands.

Definition 1: T refers to the set of all tenants, and the trust relationship of all tenants is a multiple-to-multiple relationship. For $\forall t_i, t_j \in T$, t_i is the trustor, and t_j is the trustee, which is denoted as $t_i \lhd t_j$. If t_i and t_j represent the same tenant, then $t_i \equiv t_j$.

The $\forall t_i, t_j, t_k \in T$, and TT relationship includes the following properties:

(1) Self-inspective: $t_i \lhd t_i$

(2) Inverse transmission: $t_i \lhd t_j \wedge t_i \lhd t_k \not\Rightarrow t_i \lhd t_k$

(3) Antisymmetry: $t_i \lhd t_j \wedge t_j \lhd t_i \not\Rightarrow t_i \equiv t_j$

Property 1 reflects that one tenant always trusts himself/herself, and intra-tenant access is uninfluenced by the trust relationship. Property 2 demonstrates that the trust relationship between any two tenants causes no influence on the trust relationships of another two tenants with other tenants. Property 3 reveals that one trust relationship is one-way and independent, and the mutual trust between two tenants cannot prove that the two tenants are the same tenant.

Four types of trust relationships that can realize cross-tenant access are introduced [8], namely, Type-α, Type-β, Type-γ, and Type-δ. The trust relationship of tenants involves four key problems. (1) Who is responsible for managing the trust relationship? (2) Who is responsible for authorization behavior? (3) Who provides resources? (4) Who is the authorization object? Considering $\forall t_i$, $t_j \in T$, $t_i \triangleleft t_j$, in all four types, trustor t_i is held responsible for the maintenance of trust relationship. Tab. 1 shows the differences among the four types of trust relationships.

Table 1: Types of CT trust relationships

Type of trust	Tenant in charge of authorization	Resource supplier	User provider
Type-α	t_i	t_i	t_j
Type-β	t_j	t_j	t_i
Type-γ	t_j	t_i	t_j
Type-δ	t_j	t_i	t_i

4 CT-RBAC model

To realize the fine-grained cross-tenant RBAC model for collaborative cloud services, the structure of the CT-RBAC model is designed first. Subsequently, the formal definition of the CT-RBAC model is provided, and model operations are defined. Finally, the model constraints are discussed.

4.1 CT-RBAC structure

As shown in Fig. 1, the CT-RBAC model comprises five parts, namely, tenants (T), users (U), roles (R), permissions (P), and sessions (S). Compared with the traditional RBAC model, the CT-RBAC model includes an additional tenant module and a built-in tenant attribute in U, R, and P for constructing one-to-multiple role ownership (RO) relations between roles and tenants, one-to-multiple user ownership (UO) relations between users and tenants, and one-to-multiple permission ownership (PO) relations between permissions and tenants. In other words, users, roles, and permissions all belong to one tenant in the CT-RBAC model.

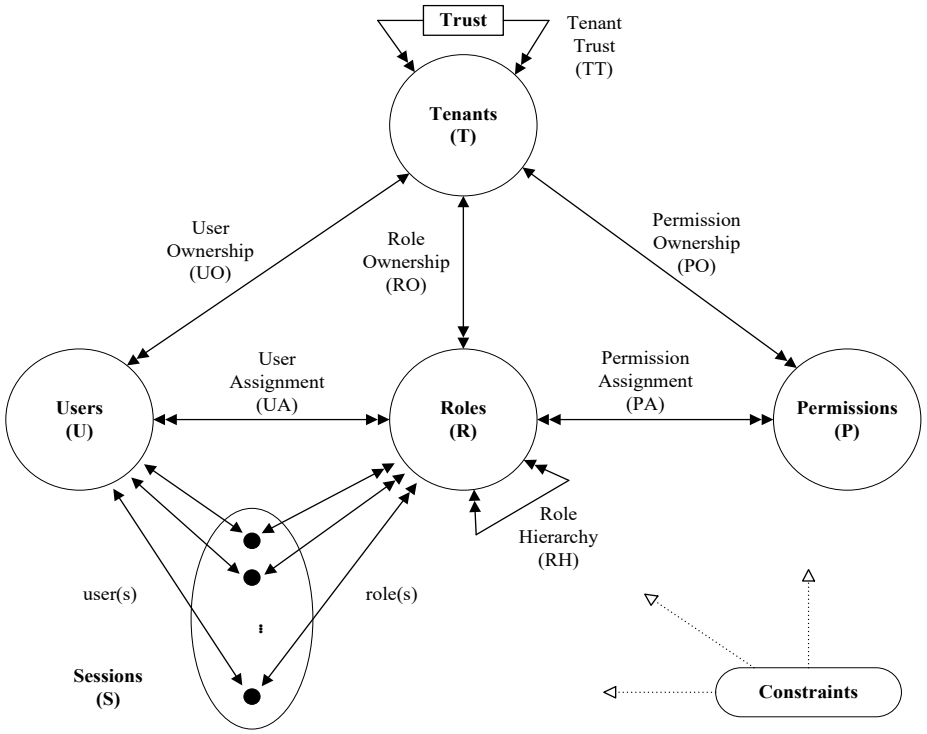

Figure 1: CT-RBAC model

Tenants (T): In the cloud computing environment, the CSP uses tenant as a logic unit to provide user storage, computing, network, and application services. T can be either an organization or a working unit. For instance, a tenant in the IaaS CSP offers 100 GB of memory space, and he/she can distribute the right to use memory space to internal users according to his/her needs. In the present study, the tenant set is denoted as T.

Users (U): User is a subject with access to resources in one tenant independently. A user might be a person, a machine, or a system. Users belong to one tenant, and one tenant can include multiple users. Here, the user set is denoted as U, and the relationship between users and tenants can be expressed by @. For example, user u_j in tenant t_i can be expressed as $t_i@u_j$.

Roles (R): One role is the object in a tenant that can implement a specific work or responsibility. R represents qualification, rights, and responsibility. R belongs to any tenant, and one tenant can cover multiple roles. The role set is denoted by R, and the relationship between roles and tenants is expressed by #. For example, role r_j in tenant t_i is expressed as $t_i\#r_i$.

Permissions (P): Permission refers to the access permission of one or multiple objects in one tenant to access in a specific mode. Permission is related with implementation details, such as reading and writing of a document in the system. Permission belongs to one tenant, and one tenant can cover multiple permissions. Here, a permission set is denoted by P, and the relationship between permissions and tenants is expressed by %. For example, permission p_j in tenant t_i can be expressed as $t_i\%p_j$.

Sessions (S): Session is a temporary activity established by one user. A session is constructed when one user activates the subset of all roles. Each session is connected with single users, and each user can be related with one or multiple sessions. Sessions about the activated roles of users in the cross-tenant cloud computing environment may be included in more than one tenant.

The trust relationship among tenants must be set up to realize their colorations. The CR-RBAC model introduces a trust relationship based on roles. Fig. 2 shows the trust relationships among tenants.

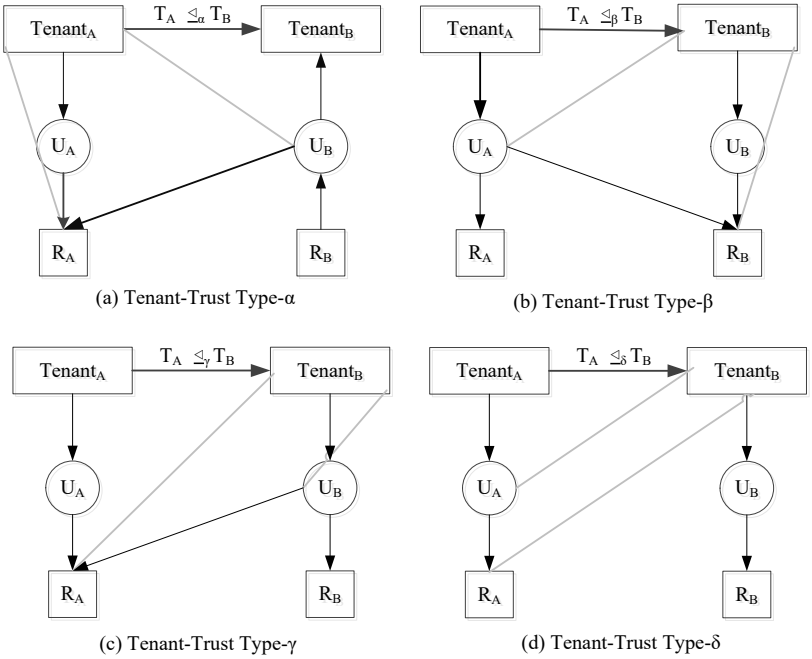

Figure 2: User-role assignment in peer-to-peer tenant-trust

4.2 Definition of models

The CT-RBAC model covers four models, namely, the CT-RBAC0, CT-RBAC1, CT-RBAC2, and CT-RBAC3 models. These four models extend the RBAC0, RBAC1, RBAC2, and RBAC3 models, which are family members of the RBAC96 model. The formal definition of the CT-RBAC0 model is introduced as follows.

Definition 2. The CT-RBAC0 model contains the following components:

T, U, R, P, S, and TT are finite sets of tenants, users, roles, permissions, sessions, and trust relationship of tenants, respectively.

- UO⊆U×T represents the mapping of relationship between each user and the tenant (multiple-to-one relationship) and is also recorded as "@". Accordingly, userOwner(u:U)→T is a function that maps the relationship between one user and the corresponding tenant. Here, userOwner(u) = t only when (u,t)∈UO.

- RO⊆R×T reflects the mapping of relationship between each role and the tenant (multiple-to-one relationship); it is denoted as "#". Accordingly, roleOwner(r:R) →T is a function that maps the relationship between one role and the corresponding tenant. In the present study, roleOwner(r) = t only when (r,t)∈RO.
- PO⊆P×T implies the mapping of relationship between one permission and the tenant (multiple-to-one relationship), and it is also denoted as "%". Accordingly, permOwner(p:P)→T is a function that maps the relationship between one permission and the corresponding tenant. permOwner(p) = t only when (p,t)∈PO.
- $PR(t_i,t_j:T)\rightarrow 2^R$ is a function that maps the role set of tenant t_i authorized to tenant t_j.
- $PU(t_i,t_j:T)\rightarrow 2^U$ is a function that maps the user set of tenant t_i authorized to tenant t_j.
- $canUse(r:R)\rightarrow 2^T$ is a function that maps the set of tenants accessible by one role. Formally, $canUse(r) = \{t\} \cup \{t_i \in T | t \triangleleft t_i \wedge r \in PR(t,t_i)\}$, where $(r,t) \in RO$.
- $canUse(u:U)\rightarrow 2^T$ is a function that maps the set of tenants accessible by one user. Formally, $canUse(u) = \{t\} \cup \{t_i \in T | t \triangleleft ti \wedge r \in PU(t,t_i)\}$, where $(u,t) \in UO$.
- UA⊆U×R is a multi-to-multi relationship between one user and one role. It reflects the role assigned by one user. (u,r)∈UA only when userOwner(u)∈canUse(r).
- PA⊆P×R is the multi-to-multi relationship between one permission and one role. It shows the permissions to assign roles. (p,r) ∈ UA only when permOwner(p) ∈ canUse(r).
- user(s:S)→U maps a function between each session and one user who is in the stated period of the session.
- $roles(s:S)\rightarrow 2^R$ is the role set in the life span of each session. roles(s) ⊆{r ∈ roles|(user(s,r) ∈UA∧userOwner(user(s))∈canUse(r)}.

The CT-RBAC0 model is the core model of CT-RBAC, and it allows setting up different types of trust relationships (Type-α, Type-β, Type-γ, and Type-δ) among different tenants.

In authorization, the trust relationship is managed by trustors. The authorization party can assign roles to users according to different types of trust relationships, but he/she is forbidden to assign roles for CTs.

Type-α: t_i must set up a trust relationship between two tenants when $\forall t_i, t_j \in T$, and $t_i \triangleleft t_j$ to realize CT access. t_j sets the PU between two tenants, t_i sets the PR between two tenants, and t_i accomplishes the authorization process.

Type-β: t_i must set up a trust relationship between two tenants when $\forall t_i, t_j \in T$, and $t_i \triangleleft t_j$ to realize CT access. t_i sets the PU between two tenants, t_j sets the PR between two tenants, and t_j accomplishes the authorization process.

Type-γ: t_i must set up a trust relationship between two tenants when $\forall t_i, t_j \in T$, and $t_i \triangleleft t_j$ to realize CT access. t_j sets the PU between two tenants, t_i sets the PR between two tenants, and t_j accomplishes the authorization process.

Type-δ: t_i must set up a trust relationship between two tenants when $\forall t_i, t_j \in T$, and $t_i \triangleleft t_j$ to realize CT access. t_i sets PU and the PU between two tenants, whereas t_j accomplishes the authorization process.

According to the analysis of the authorization of four types of trust relationships, the CT-RBAC0 model can avoid leakage of user information and role information, thereby

increasing system security. Tenants can manage the PU and PR after setting up the trust relationship. PU and PR are deleted automatically after the removal of trust relationship.

Among the RBAC96 models, RBAC1 involves a role inheritance compared with the core model. Similarly, the CT-RBAC1 model adds role inheritance relative to the CT-RBAC0 model. The inheritance of the CT-RBAC1 model covers the role inheritance in one tenant and that among different tenants. The role inheritance in one tenant in the CT-RBAC1 model follows the inheritance method of the RBAC1 model. However, the role inheritance among different tenants mainly features the following problems. (1) Who is responsible for setting up the roles? (2) Who provides the inheriting role? (3) Who offers the inherited role? (4) Who is responsible for managing role inheritance? To prevent exposure of role information, all roles in CT-RBAC1 are accomplished in the same tenant rather than among different tenants. Among the four types of trust relationships, the CT-RBAC1 model realizes CT role inheritance according to the following three rules.

Rule 1: Resource supplier (role) offers the inherited role.

Rule 2: The user who provides the access resources offers the inheriting role.

Rule 3: The authorization person is responsible for role inheritance management.

Tab. 2 presents the inheritance modes among tenants in the four types of trust relationships given $\forall t_i, t_j \in T$, and $t_i \triangleleft t_j$.

Table 2: CT role inheritance

Type of trust	Inherited role supplier	Inheriting role supplier	Manager
Type-α	t_i	t_j	t_i
Type-β	t_j	t_i	t_j
Type-γ	t_i	t_j	t_j
Type-δ	t_i	t_i	t_j

To prevent exposure of information on the inheriting role and inherited role, the set of inheriting roles and set of inherited roles that must be respectively exposed shall be established in a different tenant. The set of inherited role overlaps with the PR.

Definition 3. CT-RBAC1 inherits all components of CT-RBAC0 and meets the following conditions:

- PRH $(t_i,t_j:T)\rightarrow 2^R$ is the function of the role set in tenant t_i that can be inherited by the roles in tenant t_j.
- RH$\subseteq$R×R is the partial ordering relations on a role set and is also called role inheritance and recorded as "$\geq$"r. $r_i \geq r_j$ only when roleOwner(r_i) $\equiv$roleOwner(r_j)$\vee$ ($r_j \in PR \wedge r_i \in$PRH). If one role can inherit another role, then this role and the inherited role are either in the same tenant, or the role is in the inheritable CT role set. Meanwhile, the inherited role must be in the set of tenants that can authorize the inherited role.

- Restricted inheritance: $\forall r,r1,r2 \in R$; $r \geq r1 \cap r \geq r2 \Rightarrow r1=r2$.

The CT-RBAC1 model allows tenants to realize role inheritance under four types of trust relationships. The setting of roles at CT inheritance must be achieved by tenant managers. The inheritance modes of the four types of trust relationships are introduced as follows:

Type-α: t_i must set the PR between two tenants, whereas t_j sets the PRH between two tenants, and t_i inherits roles when $\forall t_i, t_j \in T$ and $t_i \lhd t_j$ to realize CT access.

Type-β: t_j must set the PR between two tenants, whereas t_i sets the PRH between two tenants, and t_j inherits roles when $\forall t_i, t_j \in T$ and $t_i \lhd t_j$ to realize CT access.

Type-γ: t_i must set the PR between two tenants, whereas t_j sets the PRH between two tenants and inherits roles when $\forall t_i, t_j \in T$ and $t_i \lhd t_j$ to realize CT access.

Type-δ: t_i must set the PR and PRH between two tenants, whereas t_j inherits roles when $\forall t_i, t_j \in T$, and $t_i \lhd t_j$ to realize CT access.

The RBAC2 model involves additional constraints, including static responsibility separation and dynamic responsibility separation, based on the RBAC0 model. Unlike the RBAC2 model, the constraint of the CT-RBAC2 model exists not only in the same tenant but also between tenants.

Definition 4. Static responsibility separation.

$SSD \subseteq 2^R \times N$. This relationship satisfies the following:

$\forall (rs,n) \in SSD$, and $\forall t \subseteq rs: |t| \geq n \Rightarrow \bigcap_{r \in t} assigned\_user(r) = \emptyset$,

where rs is a set of roles, t is a subset of rs, and n is a natural number higher than 2. The role set where rs lies not only covers roles in the same tenant but also roles that trust tenants can use.

Definition 5. Dynamic responsibility separation.

$DSD \subseteq 2^R \times N$. This relationship satisfies the following:

$\forall rs \in 2^R$, $n \in N$, and $(rs,n) \in D \Rightarrow n \geq 2 \cap |rs| \geq n$. Moreover,

$\forall s \in S$, $\forall rs \in 2^R$, $\forall role\_set \in 2^R$, $n \in N$, $(rs,n) \in D$, $role\_set \subseteq rs$, and $role\_set \subseteq session\_roles(s) \Rightarrow |role\_set| < n$.

Definition 6: CT-RBAC2 inherits all components of CT-RBAC0 and satisfies the following conditions:

- It conforms to the static responsibility separation.
- It conforms to the dynamic responsibility separation.

Definition 7: Inheritance ring. Among mutually trusted tenants, the role inheritance among new tenants forms a ring-shaped inheritance structure, which is called the inheritance ring.

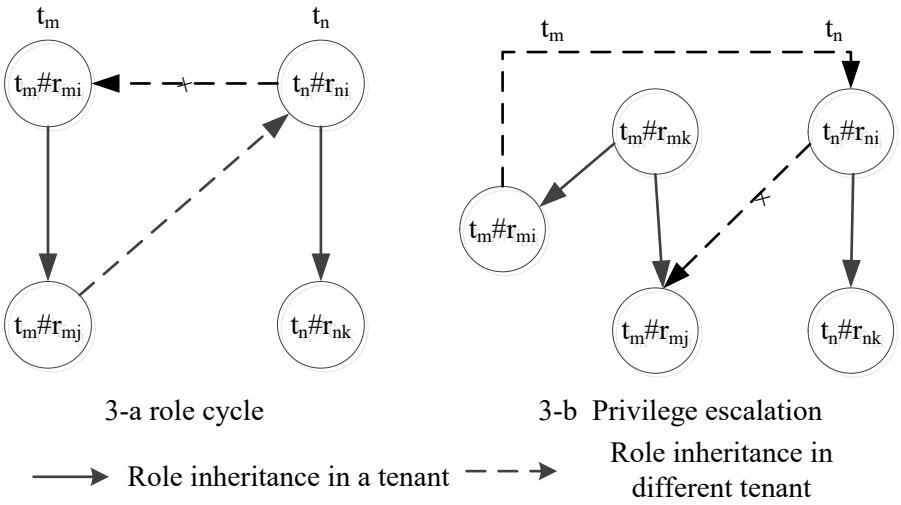

Figure 3: Conflict on role inheritance among different tenants

Fig. 3(a) shows the inheritance ring: $t_m \triangleleft t_n$ and $t_m\#r_{mi} \geq t_n\#r_{ni}$, $t_m\#r_{mj} \geq t_m\#r_{mi}$. $t_n \triangleleft t_m$ and $t_n\#r_{ni} \geq t_m\#r_{mj}$. The inheritance ring of different roles is formed due to the inheritance of trust relationships among different tenants.

Definition 8: Privilege escalation. A role gains the role accession rights, which are prohibited in the tenant, due to the role inheritance among different tenants.

Fig. 3-b illustrates the privilege escalation. $t_m \triangleleft t_n$ and $t_n\#r_{ni} \geq t_m\#r_{mi}$. $t_n \triangleleft t_m$ and $t_m\#r_{mj} \geq t_n\#r_{ni}$. For role $t_m\#r_{mj}$, a tenant gains access rights to roles $t_m\#r_{mi}$ in the same tenant due to the inheritance relationship among different tenants.

Definition 9: The CT-RBAC3 model covers CT-RBAC1 and CT-RBAC2. This model covers not only the hierarchical problems in CT-RBAC1 but also the constraint problems in CT-RBAC2. Hierarchical problems include the following:

- SSD in hierarchy: $SSD \subseteq 2^R \times N$. This relationship meets $\forall(rs,n) \in SSD$ and $\forall t \subseteq rs: |t| \geq n \Rightarrow \bigcap_{r \in t} authorized\_user(r) = \emptyset$.
- Hierarchical inheritance excludes the inheritance ring.
- Hierarchical inheritance features no privilege escalation.

The CT-RBAC0 model in the CT-RBAC model family is the basic model, and CT-RBAC1 performs hierarchical management roles compared with the basic model. CT-RBAC2 contains additional constraints relative to the basic model. CT-RBAC3 covers both CT-RBAC1 and CT-RBAC2. Specifically, CT-RBAC1 and CT-RBAC2 are incompatible. The relationships of the four models are similar with those of the family members of the RBAC96 model.

4.3 Operations of the CT-RBAC model

The operations of the CT-RBAC model mainly include the functions used by the manager of the cloud platform, the management function used by tenant managers, and

the functions used by CT managers. The functions used by CT management determine who uses the functions according to the type of trust relationships among different tenants. Tab. 3 lists the major operation functions of the models.

Table 3: Operation functions of the CT-RBAC model

Function	Condition)	Update
Management functions used by manager of the cloud platform		
AddTenant(t)	t∉T	$T'=T\cup\{t\}$
RemoveTenant(t)	t∈T	**forall** $t_e \in T$ **do** RevokeTrust(t,t_e) RevokeTrust(t_e,t) **forall** userOwner(u)≡t **do** RemoveUser(t,u) forall roleOwner(r) ≡t do RemoveRole(t,r) forall permOwner(p) ≡t do RemovePerm(t,p) T' = T'\{t}
Management functions used by the tenant managers		
AddUser(t,u)	userOwner(u)≡t∧u∉U	U'=U∪{u}
RemoveUser(t,u)	userOwner(u)≡t∧u∈U	forall {r:R|(u,r)∈UA} do RevokeUserRole(t,u,r) U'=U\{u}
AddRole(t,r)	roleOwner(r)=t∧r∉R	R' = R∪{r}
RemoveRole(t,r)	roleOwner(r)=t∧r∈R	forall {u:U|(u,r)∈UA} do RevokeUserRole(t,u,r) forall {p:P|(p,r)∈PA} do RevokeRolePerm(t,p,r) forall {r_{asc}:R(r_{asc},r)∈RH do RevokeRH(t,r_{asc},r) forall {r_{desc}:R(r_{desc},r)∈RH do

		RevokeRH(t,r_{desc},r) R' = R\{r}
AddPerm(t,p)	permOwner(p)=t∧p∉P	P'=P∪{p}
RemovePerm(t,p)	permOwner(p)=t∧p∈P	forall {r:R|(p,r)∈PA} do RevokePerm(t,p,r)
AssignUserRole(t,u,r)	t=roleOwner(r)∧u∈U	UA' = UA∪{(u,r)}
RevokeUserRole(t,u,r)	t=roleOwner(r)∧u∈U∧(u,r)∈UA	UA' = UA\{(u,r)}
AssignRolePerm(t,p,r)	t=permOwner(p)∧t∈canUse(r)	PA'=PA∪{(p,r)}
RevokeRolePerm(t,p,r)	t=permOwner(p)∧t∈canUse(r)∧(p,r)∈PA	PA'=PA\{(p,r)}
AssignRH(t,r_{asc},r)	t=roleOwner(r)∧t∈canUse(r_{asc})∧canInherit(r_{asc},r)	≥'=≥∪{(r_{asc},r)}
RevokeRH(t,r_{asc},r)	t=roleOwner(r)∧t∈canUse(r_{asc})∧r_{asc}≥r	≥'=≥\{(r_{asc},r)}
Cross-tenant access function		
AssginTrust(t,t_e)	t_e∈T	TT'=TT∪{ (t_e)}
RevokeTrust(t,t_e)	t_e∈T∧(t,te)∈TT	TT'=TT\{ (t_e)}
ExposeRole(t,t_e,r)	(t,te)∈TT∧roleOwner(r)=t	TTR'=TTR∪{(t,t_e,r)}
RevokeExposeRole(t,t_e,r)	(t,te)∈TT∧(t,t_e,r)∈TTR	TTR'=TTR\{(t,t_e,r)}
ExposeUser(t,t_e,u)	(t,te)∈TT∧userOwner(u)≡t	TTU' =TTU∪{(t,t_e,u)}
RevokeExposeUser(t,t_e,u)	(t,te)∈TT∧(t,t_e,u)∈TTU	TTU'=TTU\{(t,t_e,u)}

In cross-tenant access functions, the system can remove all access relations among tenants automatically when the tenant trust is revoked, including accessible users, accessible roles and authorization relationships of users. Similarly, the system removes authorization and inheritance of an accessible role automatically when this role is revoked from users. The rest cross-tenant access functions can be done in the same way.

5 Conclusions

Security is one of the core problems in cloud computing. In the present study, a CT-RBAC model is proposed to solve authorization problems caused by the collaboration of tenants in the cloud computing environment. At the same time, the structure, formalized definition, and operations of the model are interpreted. The proposed model possesses the characteristic of inheriting the minimum privilege principles and responsibility separation rules of the RBAC model. The model also realizes the fine-grained CT RBAC model for collaborative cloud services by the exposure of user and role information within the authorization and role inheritance constraint.

Acknowledgement: This work was sponsored by Qing Lan Project of JiangSu Province and National Natural Science Foundation of China (No. 61772280). The authors are grateful for the anonymous reviewers who made constructive comments and improvements.

References

Cohen, E.; Thomas, R. K.; Winsborough, W.; Shands, D. (2002): Models for coalition-based access control (CBAC). *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, pp. 97-106.

Curry, S.; Darbyshire, J.; Fisher, D. W. (2010): *Infrastructure Security: Getting to the Bottom of Compliance in the Cloud*. The Security Division of EMC.

Ghazizadeh, E.; Zamani, M.; Manan, J. A.; Pashang, A. (2013): A survey on security issues of federated identity in the cloud computing. *IEEE International Conference on Cloud Computing Technology & Science*.

Li, X. Y.; Shi, Y.; Guo, Y.; Ma, W. (2010): Multi-Tenancy based access control in cloud. *International Conference on Computational Intelligence and Software Engineering*, pp. 1-4.

Li, Q.; Zhang, X.; Xu, M.; Wu, J. (2009): Towards secure dynamic collaborations with group-based RBAC model. *Computers & Security*, vol. 28, no. 5, pp. 260-275.

Lin, D.; Rao, P.; Bertino, E.; Li, N.; Lobo, J. (2008): Policy decomposition for collaborative access control. *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, pp. 103-112.

Meghanathan, N. (2013): Review of access control models for cloud computing. *Computer Science & Information Science*, vol. 3, no. 1, pp. 77-85.

Nist, S. P. (2015): A NIST definition of Cloud computation. *Communications of the ACM*, vol. 53, no. 6, pp. 50-50.

Sandhu, R. (1997): Rationale for the RBAC96 family of access control models. *Proceedings of the 1st ACM Workshop on Role-Based Access Control*.

Sette, I. S.; Chadwick, D. W.; Ferraz, C. A. (2017): Authorization policy federation in heterogeneous multicloud environments. *IEEE Cloud Computing*, vol. 4, no. 4, pp. 38-47.

Tang, B.; Sandhu, R. (2013): Cross-tenant trust models in cloud computing. *Proceedings of the 14th IEEE Information Reuse and Integration*, pp. 129-136.

Tang, B.; Sandhu, R.; Li, Q. (2015): Multi-tenancy authorization models for

collaborative cloud services. *Concurrency and Computation: Practice & Experience*, vol. 27, no. 11, pp. 2851-2868.

Tang, Z.; Wei, J.; Sallam, A. (2012): A new RBAC based access control model for cloud computing. *International Conference on Advances in Grid and Pervasive Computing*, pp. 279-288.

Wang, M.; Wang, J.; Guo, L.; Harn, L. (2018): Inverted XML access control model based on ontology semantic dependency. *Computers, Materials & Continua*, vol. 55, no. 3, pp. 465-482.

Wood, T.; Ramakrishnan, K. K.; Shenoy, P.; Merwe, J. V. D.; Hwang, J. et al. (2015): CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. *ACM Transactions on Networking*, vol. 23, no. 5, pp. 1568-1583.

Yang, S. J.; Lai, P. C.; Lin, J. (2013): Design role-based multi-tenancy access control scheme for cloud services. *International Symposium on Biometrics and Security Technologies*, pp. 273-279.

Zhang, Z.; Zhang, X.; Sandhu, R. (2006): ROBAC: scalable role and organization based access control models. *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 1-9.