Role-Centric Circle of Trust in Multi-Tenant Cloud IaaS

—
—
—

Abstract. Currently, collaboration is a major challenge in adopting cloud Infrastructure-as-a-Service (IaaS), where tenants establish trust with a circle of tenants to engage in interactions with, also called as circle-of-trust. Enterprise work-flow intrinsically mandates collaboration across it's tenant boundaries as well as with associated organizations' tenants in the cloud. In this paper, we present a novel extend of role-centric access control models to provide collaboration in the context of homogeneous and heterogeneous circles. In a homogeneous circle, our approach allows tenants to equally assert cross-tenant user assignments to enable access to shared resources. In a circle with non-uniform tenants, attributes are added to distinguish user-assignments where tenants are differentiated with type in the heterogeneous circle. Particularly, tenant-trust relation is established within a group of tenants authorizing cross tenant user-assignments.

Keywords: Access Control, Circle-of-Trust, Multi-Tenant, Authorization Federation, Security.

1 Introduction

Cloud IaaS is firmly accepted by enterprises for its cost benefits, reliability, and dynamicity at scale for cloud consumers [14]. Its benefits are well documented and well practiced in the industry, but still organizations resist to fully migrate to cloud IaaS which arises from the concerns such as security, performance, uptime, and vendor lock-in. Enabling collaboration mitigates such concerns regarding vendor lock-in and different security levels required, and improves uptime and performance using different cloud providers.

In multi-tenant platforms which utilizes shared physical infrastructure, users' data are isolated into tenants to protect its privacy and integrity. In our context a tenant is regarded as an organization, a department of an organization, or an individual cloud consumer which is represented by an account in AWS [1] or a domain in OpenStack [2]. Furthermore, current cloud service providers offer federation APIs to enable collaboration between tenants such a AWS and OpenStack platforms. Besides federation between two tenants, collaboration can also be established between a set of organizations where tenants adhere to a common set of policies, trust relations and collaboration interfaces within a circle. We denote this collaboration model as a *circle-of-trust*. Scenarios such as a large enterprise with multiple tenants collaborating in a public cloud, an organization with tenants across public and private clouds, or tenants from multiple organizations performing collaborative tasks are motivating use cases for circle-of-trust.

In this paper we present novel role-based and role-centric attribute-based access control models to enable collaboration in a multi-tenant cloud IaaS circle-of-trust. Our scope of contribution is homogeneous and heterogeneous multi-tenant circles in a single cloud. We defer cross-cloud integration considerations to focus more on collaborative access control issues.

To better clarify the concept, consider the example in Figure 1 where ACME, a multinational technology corporation, aims to implement its enterprise requirements with cloud services. ACME migrates its IT infrastructure to a public cloud service provider where each tenant represents a department. ACME utilizes multiple tenants to satisfy distinct security levels required for each department. For example, Finance Dept. resources should not co-locate in the same tenant with Research & Development Dept., as Finance Dept. retains sensitive data. Furthermore, ACME organizational structure demands collaboration between its departments which is thereby required in its cloud adoption. To this end, ACME establishes a circle-of-trust among its tenants in the cloud and starts adding its tenants to the circle. For instance a new tenant created as Sales tenant in ACME, requested to join the circle. Adding additional tenants require all ACME circle members to agree on trusting the new Sales tenant. When Sales tenant joins the circle, it trusts members assertions and its assertions are likewise trusted by ACME circle members. In particular, circle-of-trust offers an association of ACME principals to collaborate in the circle.

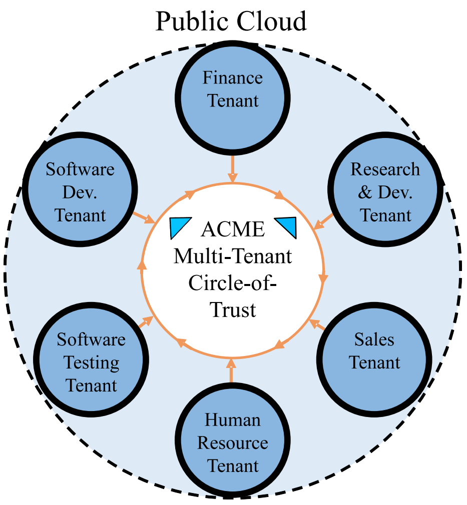


Fig. 1: ACME Corporation Multi-Tenant Circle-of-Trust.

Role-based access control (RBAC) [4, 18] and its variations has been successfully applied to cloud IaaS providing collaboration within single-cloud [19, 20] and multi-cloud systems [15]. In RBAC access permissions are assigned to roles and roles are assigned to users. Roles are central to RBAC for formulating policy and its commercial success, where it abstracts permissions into roles and role relations. With its dominance for the past two decades, RBAC limitations have been recognized leading to a push towards using attributes [6, 7, 17] with roles [11]. One method, is to add attributes to roles as role-centric attributes which takes advantage of roles' simplicity and attributes flexibility [9]. We anticipate cloud service providers will incorporate ABAC features to their current RBAC models such as role-centric to adopt convenience of RBAC with flexibility of ABAC models.

Our contribution in this paper is to design multi-tenant role-centric models with cross-tenant user-assignments. To our knowledge this is the first work considering role-centric models in circle-of-trust context.

The remainder of this paper is organized as follows. Section 1.1 overviews trust properties applicable in circle-of-trust and corresponding trust relations between tenants. In section 2, our multi-tenant role-based access control in circle denoted MT-RBAC$_c$ is proposed and specified. Section 3 introduces our multi-tenant role-centric attribute-based model in circle denoted MT-RABAC$_c$. Related work and conclusion is presented in sections 4 and 5 respectively.

1.1 Concept of Trust in Circle

In a circle-of-trust, trust relationships are defined between all circle entities. We use terms entities and principals interchangeably. Principals make assertions in the circle, assigning users to roles. Trust in the circle has the following properties, *entity coupling*, *initiation*, *direction*, and *transitivity*. Figure 2 gives a logical hierarchy of these trust properties discussed below. Vertical precedence of characteristics are arbitrary and extracted to better illustrate trust relations in our scope of contribution.

Entity Coupling (Homogeneous vs. Heterogeneous). In a circle-of trust, type of entities engaging in interactions determines homogeneity or heterogeneity of the circle shaping its authorized interactions between tenants. Moreover, with each circle type a set of trust properties are applicable. We define *homogeneous circle* where entities are uniform, for instance a circle of universities. In a homogeneous circle, collaborating principals are equally authorized to make cross-tenant authorization assertions. A *heterogeneous circle*, is an association of non-uniform entities where each type of entity is authorized specifically to make certain assertions. For instance, a circle consisting of universities, insurance companies, and banks establishes a heterogeneous circle. For example, universities can assign users to discounts in insurance companies while insurance companies cannot assign users to resources in the universities. In this paper we use type and domain interchangeably denoting the type of entities in a heterogeneous circle.
Initiation (Multilateral vs. Unilateral). If trust initiation to join a circle, is required to be confirmed by all circle members, trust is *multilateral* relation. In

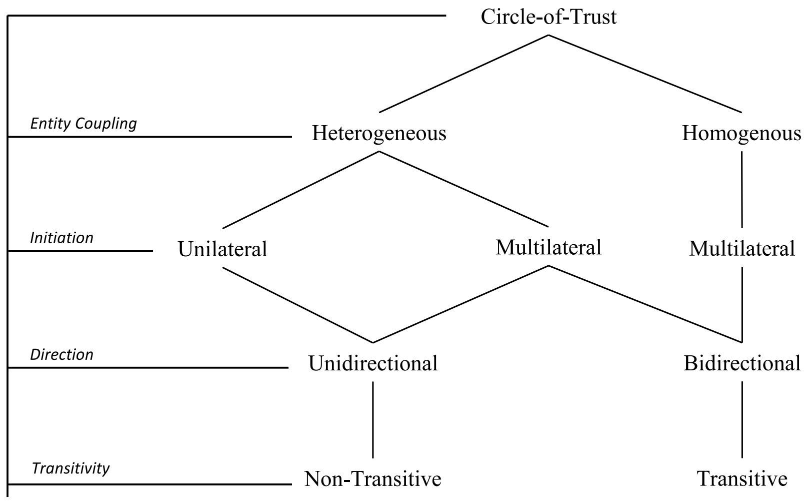


Fig. 2: Circle-of-Trust Characterization.

special situations when joining members are not authorized to make assertions (in heterogeneous circles) trust initiation is not required to be confirmed by all circle members denoted as *unilateral* relation. For instance a domain of insurance companies join a heterogeneous, unilateral circle of institutions. Insurance entities in the circle are not authorized to make assertions whilst institution entities are authorized to assert their users to discounts available to universities.
Direction (Bidirectional vs. Unidirectional). In circle trust relations, direction of trust demonstrates whether both participating circle members have equal authorizations or only one side is authorized to make assertions. If partners are authorized equally to make assertions, trust relation is *bidirectional*, otherwise it is *unidirectional* trust. Homogeneous circles' relations are bidirectional while heterogeneous circles support both trust directions. Unilateral heterogeneous circles such as given example above are only unidirectional in trust relations. Circle of universities is an example of bidirectional trust in a homogeneous circle. For example, Google services such as gmail, igoogle, and YouTube creates a heterogeneous circle where entities addition to the circle is accepted multilaterally and direction of trust is unilateral.
Transitivity (Transitive vs. Non-transitive). In a trust relation when principal "A trusts B" and "B trusts C" result in implication that "A trusts C", trust relation is denoted as transitive. In a circle, bidirectional trusts are essentially transitive where all members trust and be trusted by other circle members. In heterogeneous unidirectional circles, trust relations cannot be transitive. For example in heterogeneous unidirectional circle of institutions, banks and insurance companies, an institution can assign students to bank specific account types in banks whilst banks can assign employees to health insurances in insurance companies. Considering heterogeneous domains in the circle, a university trusting a

bank and a bank trusting an insurance entity does not necessarily imply that the university can assign students to insurance.

In this paper, we adopted multilateral, bidirectional, and transitive trust relationship for homogeneous circles. Trust relations between entities in heterogeneous circle is multilateral, unidirectional, and non-transitive relationship. In the following we identify how trust relations authorizes cross-tenant assignments in role-centric models.

1.2 Tenant-Trust in Circle

In this paper, we define trust between tenants in a circle, regarded as *tenant-trust* relationship. In a unidirectional trust relationship, common in peer-to-peer, trust is initiated and established between two tenants labeled as trustor and trustee. In a trust relation, trustor tenant is willing to trust another tenant denoted as trustee tenant. In our scope of contribution, trust is initiated multilaterally between principals in a circle. In the context of circle, trustor and trustee are not distinguished in trust relations between tenants. We identify tenants involve in a cross-tenant assignment as user-owner and resource-owner tenants. User-owner tenant owns the users in the cross-tenant assignment and resource-owner tenant owns the roles in which users are assigned to. Central to tenant-trust defined in this paper is authorizing user-owner or resource-owner tenant to assert cross-tenant assignments.

We use "$\lhd$" to represent tenant-trust where $T_A \lhd T_B$ signifies that tenant A trusts tenant B. In this relation, T_A is user-owner tenant and T_B is resource-owner tenant. Regardless of circle entity coupling, we define two types of tenant trust relations labeled as type-ϵ and type-ζ. Each tenant-trust relation type is applied to all tenants in the circle. In type-ϵ circle, user-owner tenants are authorized to assign users to roles in the circle. The following defines type-ϵ tenant-trust illustrated in Figure 3a.

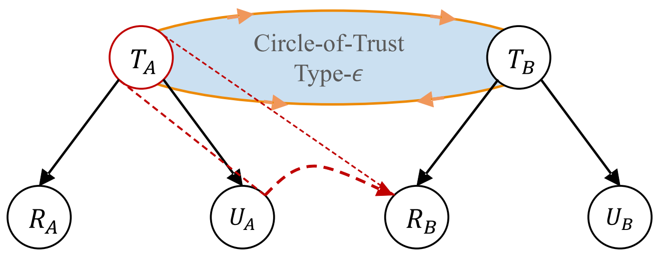
Definition 1. *If $T_A \lhd_\epsilon T_B$, then tenant T_A is authorized to assign its users to T_B's roles. Tenant T_A controls user assignments.*

In type-ζ circle, resource-owner tenants are authorized to assign users in the circle to their roles. Type-ζ is defined as the following depicted in Figure3b.

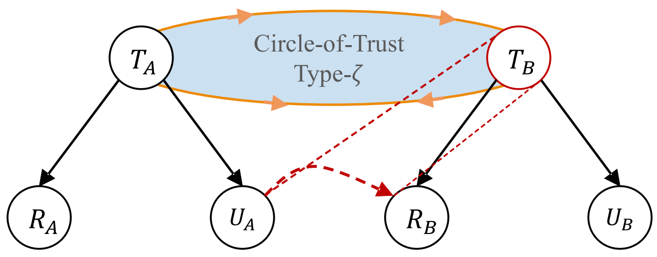
Definition 2. *If $T_A \lhd_\zeta T_B$, then tenant T_B is authorized to assign T_A's users to its roles. Tenant T_B controls user assignments.*

In homogeneous circles all peers trust each other and trust is transitive therefore $T_A \lhd T_B$ if and only if $T_B \lhd T_A$. However, in heterogeneous circles trust relations are unidirectional and non-transitive as a result $T_A \lhd T_B$ may not imply to $T_B \lhd T_A$ or vise versa.

Each tenant-trust type caters to a different security concern and objective in circle-of-trust collaboration. In type-ϵ, the objective is to merely share resources with trusted tenants where circle members are trusted with their assignments



(a) A trusts B in Circle Type-ϵ.



(b) A trusts B in Circle Type-ζ.

Fig. 3: Cross-Tenant User Assignment in Circle-of-Trust.

integrity. Resource-owner tenants can decide on resources are shared within principals in the circle using role hierarchy described in section 2. A circle of institutions is an instance of type-ϵ circle where computing resources shared between institutions. In this situation, each tenant administrator can assign its users to shared resources considering resources are not sensitive against trustor tenants' administration.

In type-ζ circle, shared resources privacy and integrity is the main concern where tenants' resources are highly sensitive data and tenants choose to control access to shared resources. Resource-owner tenants control user assignments where they can assign users in the circle to roles in their tenants to bear access. A circle of banks carries such sensitivity where banks ought to control access to their shared resources even from trusted banks within the circle. Each bank administrator assign users from tenants in the circle to its roles, enabling access to its shared resources.

2 Homogeneous Role Based Circle-of-Trust

In this section, we formally present a multi-tenant role-based access control model to enable collaboration in a homogeneous circle-of-trust which we refer to as MT-RBAC$_c$. In a homogeneous circle, tenants are equally authorized to make assertions. Collaboration in MT-RBAC$_c$ is issued through cross-tenant user assignments with respect to circle types ϵ and ζ.

2.1 Multi Tenant Role Based Circle-of-Trust (MT-RBAC$_c$)

MT-RBAC$_c$ model element sets and relations are depicted in Figure 4. It includes sets of eight basic elements: tenants (T), users (U), private roles (R_{prv}), public roles (R_{pub}), roles (R), operations (OPS), objects (OBS), and permissions ($PRMS$). In MT-RBAC$_c$ access is fundamentally defined in terms of intra-tenant and cross-tenant. Users are assigned to private roles and permissions in intra-tenant assignments. In cross-tenant assignments, users are assigned to public roles and public roles inherit private roles with limited role hierarchy. MT-RBAC$_c$ includes limited role hierarchy relation utilizing a set of public roles to create a level of abstraction which protects objects from direct assignments in a circle-of-trust.

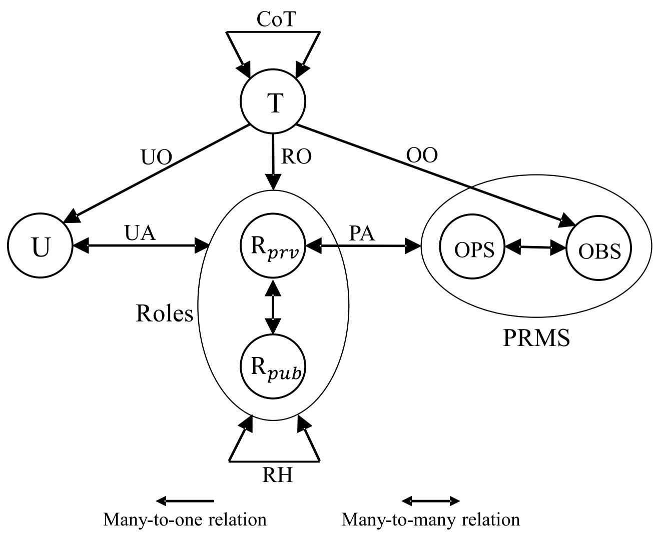


Fig. 4: Multi-Tenant RBAC Circle-of-Trust.

A *user* is a human, non-person entity, an application, or a process making requests to access objects. We scope users as a human for simplicity where $u \in U$ and U is the global set of users associated to a tenant in the cloud. In our model, a *tenant* is considered as a virtual container with tenant-specific environment for cloud services leased to cloud consumers. Practically, a tenant hosts a project, department, or an organization. Each tenant is represented as $t \in T$ where T is a global set of tenants in the cloud. Tenants are regarded as an administration domain associated to users, roles, and objects. Each user, role, and object belongs to a single tenant, called its owner, while a tenant can own multiple instances of these.

A *role* is a job function in the cloud associated with a tenant of public role and private role disjoint sets, R_{pub} and R_{prv} respectively. A *public role* is accessible by trusted tenants in a circle-of-trust. A *private role* is accessible only within its owner tenant. In contrast to private roles, public roles are not associated with permissions directly.

Resources in a tenant are represented by *objects*, each associated with an owner tenant. In our model, we consider objects as virtual machines, images, storage objects, networking $L2$ or $L3$ services, etc. Objects are owned by a tenant and paired with *operations* comprising the set of actions on cloud resources such as create, read, update, and delete. A *permission* is an authorization to perform an operation on a requested object in a tenant, such as a permission to create virtual machines in a tenant.

Central to MT-RBAC$_c$ model is tenant and role relations. Users, roles, and permissions are global sets in a cloud but they are defined per tenant with a single tenant owner. Tenant ownership relation is fundamental in how we authorize collaboration. Figure 4 illustrates *user ownership* (UO), *role ownership* (RO), and *object ownership* (OO) relations. The arrows indicate a many-to-one relation (e.g., a role can be owned by one tenant while a tenant owns many roles) stating the relation between corresponding components and owner tenant.

In Figure 4, CoT depicts circle-of-trust relation between tenants. It is defined as a many-to-many relation between tenants. As stated in section 1.2, we use "$\lhd$" to show trust relation between tenants. In a homogeneous circle, $T_A \lhd T_B$ signifies that T_A and T_B trust each other in a circle. In a homogeneous circle, each tenant-trust type-ϵ or type-ζ authorizes user-owner or resource-owner tenants. We define CoT$_\epsilon$ as follows.

Definition 3. *In a homogeneous CoT_ϵ, for all tenants t_1 where $t_1 \lhd_\epsilon t_2$, tenant t_1 is authorized to assign its users to public roles in t_2. Tenant t_1 controls cross-tenant user-role assignments of t_1's users to t_2's roles.*

In type-ϵ, user-owner tenants are authorized to assert assignments, while in type-ζ role-owner tenants are authorized. Circle-of-trust type-ζ is defined as follows.

Definition 4. *In a homogeneous CoT_ζ, for all tenants t_1 where $t_1 \lhd_\zeta t_2$, tenant t_2 is authorized to assign users from t_1 to public roles in t_2. Tenant t_2 controls cross-tenant user-role assignments of t_1's users to t_2's roles.*

In homogeneous circles, trust is transitive and tenants are equally authorized, therefore in above definitions $t \lhd_\zeta t'$ if and only if $t' \lhd_\zeta t$.

In MT-RBAC$_c$ the concept of role relations with users and permissions as illustrated in Figure 4 with *user assignment* (UA) and *permission assignment* (PA) is modified to core RBAC to reflect multi-tenancy. Fundamentally MT-RBAC$_c$ is modeled to provide collaboration with user-assignment between tenants in a circle-of-trust. To that end, user assignment is defined only if user and role owned by the same tenant or user is owned by trustee tenant in type-ϵ and trustor tenant in type-ζ. User assignment is defined only if

$$(owner\_user(u) = owner\_role(r) \wedge r \in R) \vee$$
$$(owner\_user(u) \lhd owner\_role(r) \wedge r \in R_{pub})$$

Similarly permission assignment is a many-to-many relation which assigns roles to permissions within a tenant. Permission assignment is only defined as an intra-tenant assignment therefore a permission p is assigned to a role r only if

$$(owner\_role(r) = owner\_object(o) \wedge r \in R_{prv})$$

In order to provide a secure collaboration with respect to tenants' authority on shared resources in a circle, we utilized limited role hierarchy. With two disjoint sets of private and public roles, permissions are revealed to tenants in the circle only through public roles. This arrangement provides granularity of permission to role assignment and user to role assignment. We define limited *role hierarchy* as a partial order on roles with conditional inheritance. We use "$\succeq$" to show role inheritance relation where a parent role inherits child's role permissions with following conditions.

– Private roles can inherit private roles only if both owned by the same tenant.
$$(r_1, r_2 \in R_{prv} \wedge r_1 \succeq r_2) \Rightarrow (owner\_role(r_1) = owner\_role(r_2))$$
– Private roles cannot inherit public roles.

– Public roles can inherit private roles only if both owned by the same tenant.
$$(r_1 \in R_{pub} \wedge r_2 \in R_{prv} \wedge r_1 \succeq r_2) \Rightarrow (owner\_role(r_1) = owner\_role(r_2))$$
– Public roles can inherit public roles from trusted tenants in the circle[1].
$$(r_1, r_2 \in R_{pub} \wedge r_1 \succeq r_2) \Rightarrow (owner\_role(r_1) = owner\_role(r_2) \vee$$
$$owner\_role(r_1) \lhd_{\epsilon} owner\_role(r_2) \vee owner\_role(r_2) \lhd_{\zeta} owner\_role(r_1))$$

Essentially private roles inherit private roles in a tenant whereas public roles can inherit private and public roles in a tenant and trusted tenants' public roles.

In our model, *assigned_user_roles* function gives the roles assigned to each user in the circle. The permissions assigned to each role in the model is defined *assigned_permissions* function. A user is assigned to a set of roles and each role is assigned to a set of permissions given by *authorised_user_permissions* function in MT-RBAC$_c$. We summarize the above in the following formal definition.

Definition 5. *MT-RBAC$_c$ is defined with the following basic component sets and functions.*

– *T, U, R, OPS, and OBS (tenants, users, roles, operations, and objects, respectively).*
– *$t \in T$, $u \in U$, $r \in R$, $op \in OPS$, and $ob \in OBS$.*
– *R_{pub} is a set of public roles and R_{prv} is a set of private roles where $R_{pub} \subseteq R$, $R_{prv} \subseteq R$, and $R_{pub} \cap R_{prv} = \emptyset$, $R_{pub} \cup R_{prv} = R$*
– *$PRMS = OPS \times OBS$, the set of permissions.*[2]
– *$UO \subseteq U \times T$, a many-to-one user-to-tenant owner relation.*
– *$RO \subseteq R \times T$, a many-to-one role-to-tenant owner relation.*
– *$OO \subseteq OBS \times T$, a many-to-one object-to-tenant owner relation.*

[1] Trust is defined as a reflexive relation consequently each tenant trusts itself.
[2] This is slightly different from NIST standard model where $PRMS = 2^{(OPS \times OBS)}$, and more appropriate for our purpose.

- *$owner\_user : (u : U) \rightarrow T$, the mapping of user u into its owner tenant. Formally: $owner\_user(u) = t$ where $(u, t) \in UO$.*
- *$owner\_role : (r : R) \rightarrow T$, the mapping of role r into its owner tenant. Formally: $owner\_role(r) = t$ where $(r, t) \in RO$.*
- *$owner\_object : (ob : OBS) \rightarrow T$, the mapping of object ob into its owner tenant. Formally: $owner\_object(ob) = t$ where $(ob, t) \in OO$.*
- *$CoT \subseteq T \times T$, is a many-to-many reflexive, symmetric, and transitive relation on T called circle-of-trust relation, written as $\lhd$ where $t_1 \lhd t_2$ (is equal to $t_2 \lhd t_1$) only if both t_1 and t_2 belong to same circle-of-trust.*
- *$UA \subseteq U \times R$, a many-to-many mapping user-to-role assignment relation requiring that $(u, r) \in UA \Rightarrow (owner\_user(u) = owner\_role(r) \wedge r \in R_{prv}) \vee ((owner\_user(u) \lhd_\epsilon owner\_role(r) \vee owner\_role(r) \lhd_\zeta owner\_user(u)) \wedge r \in R_{pub})$.*
- *$PA \subseteq PRMS \times R$, a many-to-many mapping permission-to-role assignment relation requiring that $((op, ob), r) \in PA \Rightarrow (owner\_object(ob) = owner\_role(r) \wedge r \in R_{prv})$.*
- *$RH \subseteq R \times R$ is a partial order on R called hierarchy relation, written as $\succeq$, where $r_1 \succeq r_2$ requiring that $(r_1, r_2) \in RH \Rightarrow ((owner\_role(r_1) = owner\_role(r_2)) \wedge \neg(r_1 \in R_{prv} \wedge r_2 \in R_{pub})) \vee ((owner\_role(r_1) \lhd_\epsilon owner\_role(r_2) \vee owner\_role(r_2) \lhd_\zeta owner\_role(r_1)) \wedge (r_1, r_2 \in R_{pub}))$.*
- *$trusted\_tenants : (t : T) \rightarrow 2^T$, the mapping of a tenant t to a set of trusted tenants in circle-of-trust. Formally: $trusted\_tenants(t) = \{t' \in T | t \lhd t'\}$*
- *$authorized\_roles : (t : T) \rightarrow 2^R$, the mapping of a tenant t to a set of authorized roles in circle-of-trust. Formally: $authorized\_roles(t) = \{r \in R \mid role\_owner(r) = t \vee role\_owner(r) \in trusted\_tenants(t)\}$*
- *$assigned\_user\_roles : (u : U) \rightarrow 2^R$, the mapping of user u into a set of roles. Formally: $assigned\_user\_roles(u) = \{r \in R \mid (u, r) \in UA\}$.*
- *$assigned\_permissions : (r : R) \rightarrow 2^{PRMS}$, the mapping of role r into a set of permissions. Formally: $assigned\_permissions(r) = \{p \in PRMS \mid (p, r) \in PA\}$.*
- *$authorized\_user\_permissions : (u : U) \rightarrow 2^{PRMS}$, the mapping of user u into a set of permissions. $authorized\_user\_permissions(u) = \bigcup_{r \in assigned\_user\_roles(u)} assigned\_permissions(r)$.*

To better clarify the concept, we present a use-case adopting MT-RBAC$_c$ in Figure 5. A group of institutions seeks to collaborate on a cyber security research program where researchers across these universities must have access to dispersed data and computing resources, and be able to collaborate on cross-projects within an association of universities. Universities establish a homogeneous Cyber circle. Each collaborating research institute become a member of Cyber circle which is homogeneous type-ϵ and partners are equally authorized to make user assignment assertions. In this scenario researchers can be assigned to research data, computation resources. In Figure 5, UTSA doesn't share its private VMs in the circle with assigning it to private roles only. Assignments are administered by user-owner tenants consistently within the circle. UTA, UTSA, UTD are tenants representing respective organizations in a public cloud platform establishing Cyber circle.

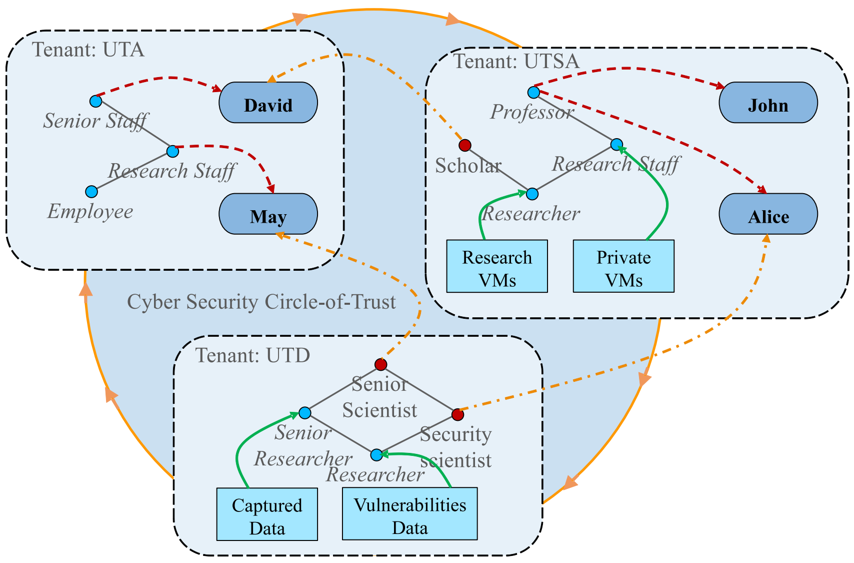


Fig. 5: Example of Multi-Tenant RBAC$_c$ in Homogeneous circle-of-Trust.

3 Heterogeneous Role and Attribute Based Circle-of-Trust

This section, presents a formal multi-tenant role-centric attribute-based access control model designated as MT-RABAC$_c$, providing collaboration in a heterogeneous circle-of-trust. Our model is motivated by previously defined role-centric model [9]. In a heterogeneous circle, entities are not equally authorized to make assertions; each tenant is authorized with respect to its tenant type (or tenant domain). In MT-RABAC$_c$, cross-tenant user assignments are conditional with respect to tenant domain attribute. Attributes are name:value pairs presenting entities' properties in the cloud.

3.1 Multi Tenant Role Centric Attribute Based Circle-of-Trust (MT-RABAC$_c$)

MT-RABAC$_c$ adds attributes to enforce cross-tenant assignment separation. Attributes are used to invoke tenant types where tenants are authorized to assert cross-tenant user assignments on certain type of tenants.

Figure 6 depicts elements in MT-RABAC$_c$, *tenant attributes* ($TATT$), *user attributes* ($UATT$), and *object attributes* ($OATT$) are added to the tenant, user, and object components respectively. *Attribute* is considered as a function which takes tenant, user, or object as input and return a value from its range. For example, an atomic-valued user attribute function such as *employeeType* returns employee status of a user $john$ where *emplyeeType* $\in UATT$, $john \in U$ and *emplyeeType*$(john) = full\_time$. Range or scope of an attribute is a finite set of atomic values specifying the valid range of attribute functions. Attribute functions either return a single value or set of values which it refers to as *atomic-valued* and *set-valued* attribute types. Besides required attributes, we expect security architects to specify attributes at system creation or modification time.

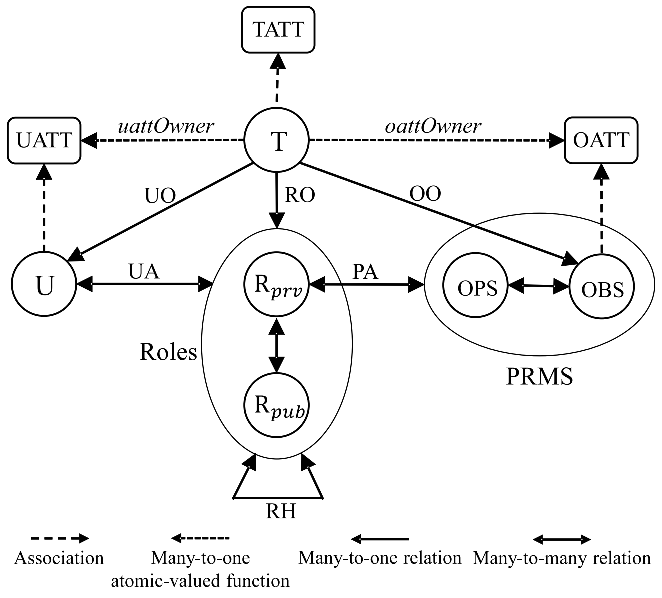


Fig. 6: Multi-Tenant Role-Centric ABAC Circle-of-Trust.

In MT-RABAC$_c$, users and objects are associated with attributes $UATT$ and $OATT$. Each user is assigned a finite set of user attributes such as name, position, salary, etc. Each user attribute is uniquely owned by a tenant, depicted by required meta-attribute *uattOwner*. User attributes are defined as partial function where a user u can be assigned a value for *uatt* only if

$$user\_owner(u) = uattOwner(uatt)$$

We defined user and object attributes per tenant to eliminate attribute conflict in presence of multi-tenancy. Each object is associated with set of $OATT$ representing properties such as creation time, risk level, VM owners, etc. Each *oatt* is owned uniquely by a tenant. We defined *oattOwner* as a required atomic-valued meta-attribute mapping *oatt* to attribute owner tenant respectively. Object attributes are similarly defined as partial function defined only if

$$user\_owner(u) = uattOwner(uatt)$$

User and object attributes can be used by security administrators to further customize access to shared resources, such as a condition where users with management position attribute can only be assigned to management roles in trusted tenants.

Tenant attributes are fundamental to MT-RABAC$_c$ to enforce limitations on cross-tenant user assignments. Moreover, we defined *domain* (D) as a set of tenant types in the system. Particularly a tenant is related to a domain with an atomic-valued required attribute function *tenantDomain*. It specifies type of a tenant. In order to separate user-assignments, *trustedDomains* is defined as a required set-valued tenant attribute specifying group of tenants which can assert assignments. In this context, user-assignment is modified with respect to trustedDomains attributes. In type-ϵ circle, user-owner tenant can assign its users to roles in tenants which it is a member of trustedDomains set. In typeζ, user assignment is modified to satisfy the condition where role-owner tenant can assign users from tenants in the circle, if it is member of their trustedDomains set. A user is assigned to a role only if

$$\begin{gathered}(owner\_user(u) = owner\_role(r) \wedge r \in R) \vee \\ (owner\_user(u) \lhd_{\epsilon} owner\_role(r) \wedge r \in R_{pub} \wedge \\ tenantDomain(owner\_user(u)) \in trustedDomains(owner\_role(r))) \vee \\ (owner\_user(u) \lhd_{\zeta} owner\_role(r) \wedge r \in R_{pub} \wedge \\ tenantDomain(owner\_role(r)) \in trustedDomains(owner\_user(u)))\end{gathered}$$

In a circle-of-trust we allow only one trust type in the circle. We don't allow both type ϵ and ζ at once to a circle due to conflict of interest. Permission-assignment remains unchanged where a permission is assigned to a role only if

$$(owner\_role(r) = owner\_object(o) \wedge r \in R_{prv})$$

Authorized_user_permisisons remains unchanged as separation already applied as user-assignments' conditions.

Definition 6. *Multi-tenant role-centric ABAC$_c$ is defined by the following enhancements and modifications to MT-RBAC$_c$.*

- *T, U, R_{prv}, R_{pub}, OPS, and OBS are defined as in MT-RBAC$_c$.*
- *D represents a finite set of existing domains.*
- *$TATT$, $UATT$, and $OATT$ represent finite set of tenant, user, and object attribute functions respectively.*
- *For each att in $TATT \cup UATT \cup OATT$, $Scope(att)$ represents the attribute's scope, a finite set of atomic values.*
- *$attType : TATT \cup UATT \cup OATT \rightarrow \{set, atomic\}$, specifies attributes as set or atomic valued.*
- *UO, RO, and OO represents user, role, and object owner many-to-one relations as defined in MT-RBAC$_c$.*
- *owner_user, owner_role, and owner_object are functions mapping users, roles, and objects to owner tenant respectively, defined in MT-RBAC$_c$ model.*

- *Each tenant attribute function maps elements in T to atomic or set values as follows.*

  $\forall tatt \in TATT.tatt : T \to \begin{cases} Scope(tatt)\ if\ attType(tatt) = atomic \\ 2^{Scope(tatt)}\ \ if\ attType(tatt) = set \end{cases}$
- *Each user attribute function $uatt \in UATT$ is defined as a partial function mapping elements in U to atomic or set values.*

  $\forall uatt \in UATT.uatt : U \hookrightarrow \begin{cases} Scope(uatt)\ if\ attType(uatt) = atomic \\ 2^{Scope(uatt)}\ \ if\ attType(uatt) = set \end{cases}$

  $uatt(u : U)$ is defined only if $uattOwner(uatt) = owner\_user(u)$.
- *Each object attribute function $oatt \in OATT$ is defined a partial function mapping elements in O to atomic or set values.*

  $\forall oatt \in OATT.oatt : O \hookrightarrow \begin{cases} Scope(oatt)\ if\ attType(oatt) = atomic \\ 2^{Scope(oatt)}\ \ if\ attType(oatt) = set \end{cases}$

  $oatt(o : O)$ is defined only if $oattOwner(oatt) = owner\_object(o)$.
- *$MATT = \{uattOwner, oattOwner\}$, required meta-attribute functions.*
  - *$uattOwner : (uatt : UATT) \to T$, required user atomic meta attribute function, mapping user attribute uatt to attribute owner tenant t.*
  - *$oattOwner : (oatt : OATT) \to T$, required object atomic meta attribute function, mapping object attribute oatt to attribute owner tenant t.*
- *$tenantDomain : (t : T) \to D$, required tenant atomic attribute function mapping tenant t to tenant domain d. $tenantDomain \in TATT$.*
- *$trustedDomains : (t : T) \to 2^D$, required tenant set attribute function mapping tenant t to powerset of trusted domains D. $trustedDomains \in TATT$.*
- *$UA \subseteq U \times R$, a many-to-many mapping user-to-role assignment relation requiring that $(u, r) \in UA \Rightarrow (owner\_user(u) = owner\_role(r) \wedge r \in R) \vee (owner\_user(u) \lhd_{\epsilon} owner\_role(r) \wedge r \in R_{pub} \wedge\ tenantDomain(owner\_user(u)) \in trustedDomains(owner\_role(r))) \vee\ (owner\_user(u) \lhd_{\zeta}\ owner\_role(r) \wedge r \in R_{pub} \wedge\ tenantDomain(owner\_role(r)) \in trustedDomains(owner\_user(u)))$.*
- *$PA \subseteq PRMS \times R$, a many-to-many mapping permission-to-role assignment relation requiring that $((op, ob), r) \in PA \Rightarrow (owner\_object(ob) = owner\_role(r) \wedge r \in R_{prv})$.*

4 Related Work

The Liberty Alliance Project [21] identified the conceptual framework and guidelines in a circle-of-trust as part of their federated identity vision. Considerable research on circle-of-trust has been devoted to identity federation such as Kylau et al. [12] trust requirements and patterns in circle-of-trust identity federation. Also, Boursas et al. [3] presented circle-of-trust collaboration trust considerations in identity federation for assessment of entities' trust outside the circle. The previous research in circle-of-trust has concentrated on identity federation whereas our scope of contribution lies within authorization federation.

RBAC [4] has been considerably extended to enable collaboration. ROBAC [22] extended RBAC to consider authorization in multiple organizations, however the collaboration within organizations is not considered. In [10], authors proposed a group-centric framework to share resources within a group where collaboration is enabled over the group, as well as GB-RBAC [13] which extends RBAC with groups to support collaboration. Delegation models such as d-RBAC [5] proposed delegation of roles but delegation relations are not flexible enough in cloud systems.

Further in cloud IaaS, models such as CTTM [19] extended RBAC to enable collaboration in multi-tenant cloud systems. In [15], authors presented cross-tenant collaboration models in multi-cloud environments. Recently, there has growing interest in attribute-based [8] and its integrations with roles [11]. Jin et al. [9] added attributes to roles where role permissions are constrained by attributes using permission filtering, this approach utilizes role-centric attribute model. In ABAC collaboration, MT-ABAC [16] is proposed where collaboration is enabled through cross-tenant attribute assignment in multi-tenant cloud IaaS. Above mentioned research, studied peer-to-peer collaboration however we are realizing collaboration in the context of circle-of-trust in multi-tenant cloud IaaS.

5 Conclusion

This paper provided a fine-grained collaboration model in a circle-of-trust. We presented the MT-RBAC$_c$ model in a homogeneous circle, in which collaboration is enabled through user to public role assignments. We identified, private and public roles with limited role hierarchy to control access on tenants' resources. Trust is defined on tenants with types ϵ and ζ (denoting as circle type) authorizing user-tenant and resource-tenant assertions respectively. Moreover, attributes were added in MT-RABAC$_c$ to classify tenants into domains in heterogeneous circles, where tenant-trust is defined conditionally with *trustedDomain* attributes. Using roles and attributes to enable cross-tenant user-assignments makes our approach general and dynamic enough to address current issues while it is applicable to current platforms. For future work, we plan to extend this work with attribute-based models into further generalization in multi-cloud environments. Finally, implementing proposed models within current cloud platforms is another next step.

Acknowledgement

This research is partially supported by *anonymized.*

References

1. Amazon AWS. `https://aws.amazon.com/`.
2. OpenStack. `http://www.openstack.org/`.
3. L. Boursas and V. A. Danciu. Dynamic inter-organizational cooperation setup in circle-of-trust environments. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 113–120. IEEE, 2008.

4. D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *TISSEC*, 4(3):224–274, 2001.
5. E. Freudenthal, T. Pesin, et al. dRBAC: distributed role-based access control for dynamic coalition environments. In *Proc. of ICDCS*, pages 411–420. IEEE, 2002.
6. V. C. Hu, D. Ferraiolo, et al. Guide to attribute based access control (ABAC) definition and considerations. *NIST Special Publication*, 800:162, 2014.
7. V. C. Hu, D. R. Kuhn, and D. F. Ferraiolo. Attribute-based access control. *Computer*, (2):85–88, 2015.
8. X. Jin, R. Krishnan, and R. S. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. *DBSec*, 12:41–55, 2012.
9. X. Jin, R. Sandhu, and R. Krishnan. RABAC: role-centric attribute-based access control. In *Computer Network Security*, pages 84–96. Springer, 2012.
10. R. Krishnan, R. Sandhu, J. Niu, and W. Winsborough. Towards a framework for group-centric secure collaboration. In *Collaborative Computing: Networking, Applications and Worksharing, 2009. CollaborateCom 2009. 5th International Conference on*, pages 1–10. IEEE, 2009.
11. D. R. Kuhn, E. J. Coyne, and T. R. Weil. Adding attributes to role-based access control. *Computer*, (6):79–81, 2010.
12. U. Kylau, I. Thomas, M. Menzel, and C. Meinel. Trust requirements in identity federation topologies. In *Advanced Information Networking and Applications, 2009. AINA'09. Int. Conf. on*, pages 137–145. IEEE, 2009.
13. Q. Li, X. Zhang, M. Xu, and J. Wu. Towards secure dynamic collaborations with group-based RBAC model. *Computers & Security*, 28(5):260–275, 2009.
14. P. Mell and T. Grance. The NIST definition of cloud computing. 2011.
15. N. Pustchi, R. Krishnan, and R. Sandhu. Authorization federation in IaaS multi cloud. In *Proc. of Security in Cloud Computing*, pages 63–71. ACM, 2015.
16. N. Pustchi and R. Sandhu. Mt-abac: A multi-tenant attribute-based access control model with tenant trust. In *Network and System Security*, pages 206–220. Springer, 2015.
17. R. Sandhu. The authorization leap from rights to attributes: maturation or chaos? In *Proc. of SACMAT*, pages 69–70. ACM, 2012.
18. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.
19. B. Tang and R. Sandhu. Cross-tenant trust models in cloud computing. In *Proc. of Int. Conf. IRI*, pages 129–136. IEEE, 2013.
20. B. Tang, R. Sandhu, and Q. Li. Multi-tenancy authorization models for collaborative cloud services. In *Proc. of CTS*, pages 132–138. IEEE, 2013.
21. T. Wason, S. Cantor, J. Hodges, J. Kemp, and P. Thompson. Liberty id-ff architecture overview. *Liberty Alliance*, 2004.
22. Z. Zhang, X. Zhang, and R. Sandhu. ROBAC: Scalable role and organization based access control models. In *Proc. of CollaborateCom*, pages 1–9. IEEE, 2006.