Cross-Tenant Trust Models in Cloud Computing

Bo Tang and Ravi Sandhu
Institute for Cyber Security
and Department of Computer Science
University of Texas at San Antonio
One UTSA Circle, San Antonio, Texas 78249
Email: btang@cs.utsa.edu, ravi.sandhu@utsa.edu

Abstract

Most cloud services are built with multi-tenancy which enables data and configuration segregation upon shared infrastructures. Each tenant essentially operates in an individual silo without interacting with other tenants. As cloud computing evolves we anticipate there will be increased need for tenants to collaborate across tenant boundaries. This will require cross-tenant trust models supported and enforced by the cloud service provider. Considering the on-demand self-service feature intrinsic to cloud computing, we propose a formal cross-tenant trust model (CTTM) and its role-based extension (RB-CTTM) integrating various types of trust relations into cross-tenant access control models which can be enforced by the multi-tenant authorization as a service (MTAaaS) platform in the cloud.

1. Introduction

Cloud computing has developed rapidly and become a force transforming the IT industry. Its service models have been increasingly accepted by consumers and enterprises. The on-demand self-service feature and the trend towards integrating computing resources in the cloud bring new challenges to fine-grained access control mechanisms, since the existing models from traditional environments are either incompatible or inefficient.

The service model of cloud computing is intrinsically a self-service outsourcing model. A cloud consumer outsources part of its computing resources to a cloud service provider (CSP). The consumer can unilaterally manage its data and settings through interfaces given by the CSP as well as its share of computing resources. The CSP is responsible enforcing appropriate access control mechanisms to consumer data and resources in a multi-tenant manner.

Multi-tenancy is a basic feature of cloud computing. It seeks to isolate activities of tenants from each other to protect data security and privacy. Thus, intra-tenant access control can be simply aligned with traditional single-domain models, such as role-based access control (RBAC) [16, 27]. But cross-tenant access control cannot be appropriately achieved by traditional models. Currently many CSPs simply block cross-tenant accesses in the cloud. This solution raises many problems, such as data lock-in [6], which restrict the development of cloud computing. In order to break the barrier between tenants in a controllable way, a suitable fine-grained cross-tenant access control model is essential.

Currently, Single Sign-On (SSO) techniques are combined with Security Assertion Markup Language (SAML) to achieve authentication and simple authorization in federated cloud environments [3], but fine-grained authorization is not supported. RBAC has been integrated into Nebula [23], a private cloud system, by NASA to enable fine-grained authorization. However, it supports only a centralized authority which does not exist in public clouds. IBM [14] and Microsoft [13] proposed a resource sharing approach in data-centric clouds using database schema which is only applicable to databases. Some role-based trust relations have been introduced [11, 19, 28, 29] to bridge the gap between authorization domains. While these approaches are suitable for specific aspects of cloud computing, there remains need for a general model of cross-tenant access control.

In this paper, we propose a cross-tenant trust model (CTTM) which encompasses various types of trust relations bridging authorization domains of each tenant. We argue that cross-tenant trust relation should be reflexive, but not transitive, symmetric or anti-symmetric. We identify four potential types of cross-tenant trust, and argue that only three of them are viable. Based on the cross-tenant trust relations we present formalized models of CTTM and a role-based extension (RB-CTTM). Furthermore, we propose a multi-tenant authorization as a service (MTAaaS) platform enforcing multi-tenant access control mechanisms in the cloud.

The rest of this paper is organized as the following. Section 2 explains the benefit of trust relations in facilitating cross-tenant access control in the context of cloud computing given a familiar example from the car rental business. The essential characteristics of the cloud and the scopes and assumptions of this paper are also discussed. Section 3 presents an analysis and classification of cross-tenant trust relations, a formalization of CTTM and RB-CTTM, and a conceptual design of the MTAaaS platform. In Section 4 we discuss related work in cross-domain and multi-tenant access control models. Section 5 concludes this work.

2. Background and Motivation

In a social context, trust has several connotations. Definitions of trust typically refer to a situation characterized by the following aspects. One party (trustor) is willing to rely on the actions of another party (trustee) with respect to the future. In addition, the trustor (voluntarily or forcedly) abandons control over the actions performed by the trustee [2]. This definition of trust is also applicable in the virtual world, including cloud computing. For example, cloud consumers trust cloud providers to manage their data while cloud providers trust cloud consumers to use their computing resources responsibly. These two trust relations are both established by a service level agreement (SLA) which regulates the responsibilities of each party.

2.1 Motivation

The trust relation between two tenants of a cloud service provider is analogous to the trust relation between a car rental company, say AVIS, and a customer organization, say UTSA. Let $AVIS$ and $UTSA$ represent two tenants in a Platform as a Service (PaaS) [24] for AVIS and UTSA respectively. The PaaS is in charge of hosting applications for its tenants. Figure 1 illustrates an example of cross-tenant access needs between these two tenants.

Assume AVIS and UTSA have an agreement about discounted car rental price from AVIS exclusively for UTSA students. The agreement itself is an established trust relation created outside of the PaaS. In traditional practices, AVIS may give away coupons on UTSA campus or send coupon code to UTSA mailing lists. These approaches provide little control to the distribution of coupons and their use. Thus, controlling access of the discount privilege in the cyber domain is critical in the overall trust relationship.

In this example, the user information of UTSA is stored in the cloud, more specifically in the PaaS, where the discount privilege of AVIS can also be accessed. Thus, the access control mechanisms of cross-tenant accesses from $UTSA$ users to $AVIS$ permissions are provided by the PaaS. For instance, Bob as a student user in $UTSA$ wants to access the discount permission in $AVIS$, as shown in Figure 1. In order to securely enable this cross-tenant access, the PaaS should support an appropriate trust model regulating who builds the trust and who executes the trust.

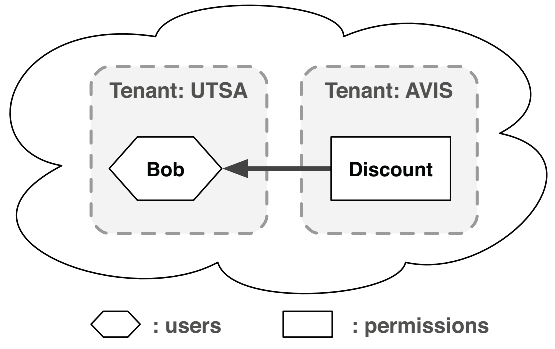

Figure 1. A car rental example of cross-tenant access.

2.2 On-Demand Self-Service in the Cloud

On-demand self-service is one of the essential characteristics of the cloud [24]. CSPs provide centralized facilities of computing resources which are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. A consumer can unilaterally provision the computing resources as needed automatically without human interaction with the CSP. Moreover, the tenants and users can only be treated as temporary entities since a user can easily create a cloud user account, get a tenant in a cloud service for some tasks and release the tenant when the job is done. The user account may also be removed after usage.

In cross-tenant accesses, the self-service feature requires agility in the trust model. The trustor and the trustee may be created on-demand so that the trust relation between them should be established and destroyed easily. Additionally, a trust negotiation process for a bilateral trust may not be suitable in this environment. Instead, unilateral trust relations asserted by the trustor better match the on-demand self-service feature of the cloud.

2.3 Scope and Assumptions

In this paper, we adopt the following assumptions.

Standardized APIs. The CSP should have a set of standardized APIs and other necessary facilities, in order to functionally enable cross-tenant accesses. Our research mainly focusses on the access control of these APIs.

Authenticated Users. All the users requesting accesses are assumed to have been properly authenticated.

These two assumptions are more or less necessary for any treatment of cross-tenant trust. The rest of the assumptions below could be relaxed or removed but are convenient for an initial investigation of cross-tenant trust models.

One Cloud Service. For simplicity, we assume cross-tenant accesses are between tenants of a single cloud service. We believe our models are extensible beyond a single cloud but multi-cloud considerations are outside the scope of this paper.

Two Tenant Trust. For simplicity we only consider trust relations between two tenants. More generally, there may exist trust relations for more than two tenants forming a community, a coalition, or a federation [9].

Unidirectional Trust Relations. Each trust relation is unidirectional (like follow in Twitter) as opposed to bidirectional (like friend in Facebook).

Unilateral Trust Relations. Each trust relation is established unilaterally by the trustor, and remains under exclusive control of the trustor. Specifically, the trustor and only the trustor can create, modify, and revoke a trust relation. In general, it seems unreasonable for the trustee to unilaterally assert a trust relationship. However, it may be reasonable for a trustee to agree before a trustor can assert trust. Also, it may be reasonable for a trustee to unilaterally revoke trust with respect to a trustor. Treatment of these cases is outside the scope of this paper.

3. Cross-Tenant Trust Model

Cross-Tenant Trust Model (CTTM) consists of different types of unilateral trust relations which reflect different needs in access control between two tenants, the trustor and the trustee. In this section, we first present an analysis of the tenant trust (TT) relations and discuss their types and usage. Then, we give a formalization of CTTM and its role-based extension (RB-CTTM). Furthermore, in order to argue the feasibility of the cross-tenant trust models in the cloud, we propose a multi-tenant authorization as a service (MTAaaS) platform to facilitate the enforcement.

3.1 Tenant Trust Relations

Before we discuss the formalization of CTTM, we first give an analysis of tenant trust (TT) relations which is the crucial part of our cross-tenant access control models. TT (also written as "$\trianglelefteq$") is a binary relation from the trustor to the trustee. Let "$\equiv$" denote the same tenant relation. For example, "$A \equiv B$" means that A and B are the same tenant. Let T be the set of all tenants. For $\forall A, B, C \in T$, a TT relation is reflexive

$$A \trianglelefteq A \tag{1}$$

but not transitive

$$A \trianglelefteq B \wedge B \trianglelefteq C \nRightarrow A \trianglelefteq C \tag{2}$$

and it is neither symmetric

$$A \trianglelefteq B \nRightarrow B \trianglelefteq A \tag{3}$$

nor antisymmetric

$$A \trianglelefteq B \wedge B \trianglelefteq A \nRightarrow A \equiv B. \tag{4}$$

Statement (1) requires that a tenant always trusts itself since intra-tenant accesses are not influenced by the trust relations. In order to control the propagation of trust relations and cross-tenant accesses enabled by the trust relations, Statement (2) requires that a trust relation can only be directly defined by the trustor but is never inferred from indirect combination of other trust relations. Statement (3) and (4) basically require that a trust relation is unidirectional and independent in each direction. A single tenant can be the trustor in one trust relation and the trustee in another. Together these statements characterize the building and basic characteristics of cross-tenant trust.

Turning to the usage of TT, we identify four potential types of trust relations to enable and control cross-tenant accesses.

- Type-α: trustor can give access to trustee.
- Type-β: trustee can give access to trustor.
- Type-γ: trustee can take access from trustor.
- Type-δ: trustor can take access from trustee.

The terms of "giving" and "taking" accesses distinguish the authorities of issuing cross-tenant assignments. Sticking with the car rental example, $AVIS$ giving access to $UTSA$ is equivalent to $AVIS$ assigning $AVIS$'s permissions to $UTSA$'s users. Conversely, $UTSA$ taking access from $AVIS$ means $UTSA$ can make the same assignment.

Type-α trust (also written as "$\trianglelefteq_\alpha$") is most intuitive since it is closest to familiar real world trust relations. For example, by establishing $AVIS \trianglelefteq_\alpha UTSA$, $AVIS$ can obtain user information in $UTSA$ and assign cross-tenant accesses from $UTSA$'s users to $AVIS$'s permissions. In this type of trust relation, the trustor ($AVIS$) holds the authority of assigning its own permissions to the trustee's users and requires visibility to the trustee's ($UTSA$'s) user information. Nevertheless, user information is also considered as sensitive information for $UTSA$ and $UTSA$ may wish to limit its exposure. Note that the trust is unilaterally asserted by the trustor $AVIS$ which enables visibility into $UTSA$'s user information without any involvement of $UTSA$. In such cases, Type-α trust is not suitable.

Type-β trust (also written as "$\trianglelefteq_\beta$") alters the direction of the trust relation in Type-α trust so that the trustor ($UTSA$) can control the exposure of its user information which is necessary for the trustee ($AVIS$) to make cross-tenant authorization assignments. In the car rental example, by establishing $UTSA \trianglelefteq_\beta AVIS$, $UTSA$ explicitly exposes its user information to $AVIS$ so that $AVIS$ can assign its permissions to $UTSA$'s users based on $UTSA$'s user information. In this way, access to the discount permission is controlled by both the trustor ($UTSA$) and the trustee ($AVIS$) together. No single party can unilaterally authorize a cross-tenant access.

In Type-γ trust (also written as "$\trianglelefteq_\gamma$"), the cross-tenant access is also controlled by both the trustor and the trustee together. But it is very different than Type-β trust. By establishing the trust relation, the trustor delegates the control of cross-tenant authorization assignments to the trustee. Thus, in order to maintain the control of cross-tenant accesses, the trustor doesn't issue cross-tenant assignments but just appropriately manage the trust relations which is required for the assignments to take effect. For instance, by establishing $AVIS \trianglelefteq_\gamma UTSA$, $AVIS$ delegates $UTSA$ to assign cross-tenant access from $UTSA$'s users to $AVIS$'s permissions. Because $UTSA$ is more familiar with the organization of its user information, $UTSA$ is more knowledgable to assign its users to $AVIS$'s permissions such as discounts. For instance $UTSA$ can determine which users within UTSA get the discount, e.g., full-time students but not part-time students.

Type-δ trust relation does not provide meaningful use cases since the trustor holds all the control of the cross-tenant assignments of the trustee's permissions. For example, $UTSA$, or any other tenants in the cloud service, can unilaterally trust $AVIS$ and assign $AVIS$'s permission to its own users. In this kind of situation, cross-tenant accesses cannot be controlled by the permission owners. Thus, Type-δ trust relation has little practical usage in cross-tenant access control, and we will ignore it henceforth.

3.2 Formalized Model

In the formalization of CTTM, as shown in Figure 2, there are three entity components: *users* (U), *permissions* (P) and *tenants* (T). Both the U and P components exist in most of the formalized access control models since they are critical in expressing an access. A novel component T is introduced to express accesses in multi-tenant environments in which the other components should fit. In particular a user in U and a permission in P are owned respectively by a tenant in T so that they can be identified uniquely in a multi-tenant access in cloud environments. This is depicted

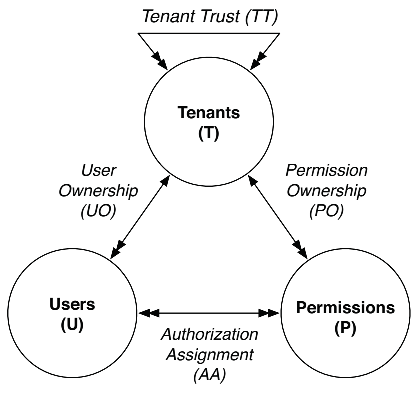

Figure 2. Cross-Tenant Trust Model

by the UO and PO relations in Figure 2. PO is a many-to-one relation from P to T.

UO may be a many-to-one relation or a many-to-many relation from U to T depending on implementation. In a many-to-many case, a user may be associated with multiple tenants out of which if the permission owner trusts one or more, then the access can be granted to the user. Here, conflict of permissions[a] may happen during the policy evaluation process because various trust relations are invoked for one user. This problem is out of the scope of this paper. For simplicity, we choose to define UO as a many-to-one relation in CTTM formalization.

TENANTS. A tenant is an exclusive virtual partition of a cloud service leased from a CSP [28]. In practice a tenant is usually mapped to a project, a department, or an organization. Cloud user activities and resource accesses are defined within the domain of a tenant. For example, $UTSA$ is a tenant created for UTSA as an organization customer of the PaaS service so that UTSA can manage its users and resources in the domain of $UTSA$ tenant[b].

USERS. A user is an identifier for an individual or a process in a tenant. Each user has a single owner tenant while a tenant has multiple users. A user is formed by a username and a tenant and is written in the format of "username@tenant". Note that an individual or a process may possess multiple users in different tenants. These users are treated independently. In the car rental example, $Bob@UTSA$ is a user acting for Bob in $UTSA$.

PERMISSIONS. A permission is a specification of a privilege in a tenant. It is formed by a permission name and a tenant and is written in the format of "permission_name%tenant". Each permission has a single owner tenant while a tenant has multiple permissions. For exam-

[a]Conflict may arise if negative permissions are allowed in the access control policy.

[b]In a more general treatment we would identify the cloud service explicitly in a two part name such as $UTSA.CloudService$.

ple, $discount\%AVIS$ represents the discount permission in $AVIS$.

The formal definition of CTTM follows.

Definition 1 *The cross-tenant trust model (CTTM) has the following components.*

- *T, U, and P are finite sets of tenants, users, and permissions respectively;*
- *$UO \subseteq U \times T$, is a many-to-one relation mapping each user to its owner tenant; correspondingly, $userOwner(u : U) \rightarrow T$, is a function mapping a user to its owner tenant where $userOwner(u) = t$ iff $(u, t) \in UO$;*
- *$PO \subseteq P \times T$, is a many-to-one relation mapping each permission to its owner tenant; correspondingly, $permOwner(p : P) \rightarrow T$, is a function mapping a permission to its owner tenant where $permOwner(p) = t$ iff $(p, t) \in PO$;*
- *$TT \subseteq T \times T$ is a many-to-many tenant trust relation on T, also written as "$\trianglelefteq$"; depending on the system TT can be one of Type-α, Type-β or Type-γ;*
- *$AA \subseteq U \times P$, a many-to-many user-to-permission assignment relation, also written as "$\leftarrow$", requiring that $u \leftarrow p$ only if*
  $permOwner(p) \equiv userOwner(u) \vee$
  $permOwner(p) \trianglelefteq_\alpha userOwner(u) \vee$
  $userOwner(u) \trianglelefteq_\beta permOwner(p) \vee$
  $permOwner(p) \trianglelefteq_\gamma userOwner(u)$,
  where only one of the $\trianglelefteq$ requirements can apply depending on the nature of TT.[c]

In Definition 1, AA represents a set of multi-tenant assignments, including cross-tenant and intra-tenant assignments. AA should comply with the conditions specified in the definition. Intra-tenant assignments are always prohibited, since $permOwner(p) \equiv userOwner(u)$ is always true and moreover TT is reflexive for each type of trust. For cross-tenant assignments, the permission owner should be the trustor either in a Type-α or a Type-γ trust relation, or the trustee in a Type-β trust relation, while the user owner is on the other end of the trust relations. For example, in order to enable the cross-tenant assignment "$Bob@UTSA \leftarrow discount\%AVIS$", the appropriate one of the following three trust relations should exist: $AVIS \trianglelefteq_\alpha UTSA$, $UTSA \trianglelefteq_\beta AVIS$, or $AVIS \trianglelefteq_\gamma UTSA$, depending on the nature of TT.

The revocation of cross-tenant authorization assignments in CTTM can be achieved in two ways. One way is revoking the assignment by the assignment issuer (executor of the trust relation) which is the trustor in Type-α and Type-β trust or the trustee in Type-γ trust. Since the tenant trust relations are required in authorizing cross-tenant accesses, the other way to revoke a cross-tenant AA is removing all the TT it depends on by the respective trustors (builder of the trust relation) who are the permission owners in both Type-α and Type-γ trust or the user owner in Type-β trust. Note that by removing a trust relation, all of the authorization assignments depending on the particular trust relation are automatically revoked[d]. Moreover, removing a trust relation does not affect intra-tenant assignments.

[c]One could consider allowing TT to include a mix of the $\trianglelefteq$ relations but this is likely to be confusing and overkill.

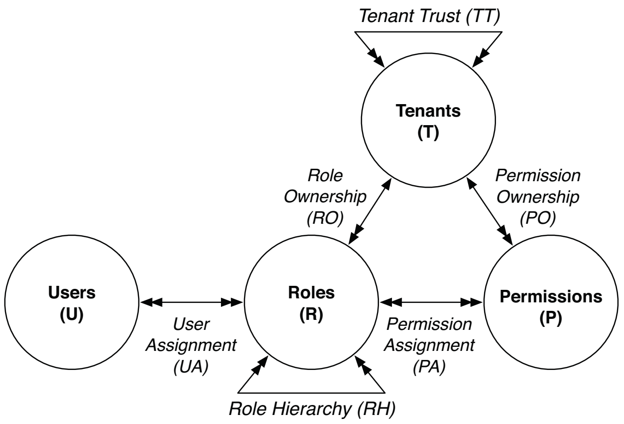

Figure 3. Role-Based Cross-Tenant Trust Model

3.3 Role-Based CTTM

Role-based access control (RBAC) models [16, 27] have been utilized by enterprise information systems for decades. The introduction of roles intermediates the authorization assignments from users to permissions and easies administration of access control policies. The benefit of roles is also applicable to CTTM. Due to the on-demand self-service feature of the cloud, managing the authorization assignments from users directly to permissions is subject to dynamic changes of users and tenants. Therefore, we propose a reasonable extension of CTTM, role-based CTTM (RB-CTTM) in which each user can have different roles in different tenants and a role belongs to a single tenant so that a change of the user does not affect the entire authorization assignment.

The RB-CTTM model, as shown in Figure 3, contains four entity components: *users* (U), *roles* (R), *permissions*

[d]For simplicity and security at the model level, we assume that the corresponding cross-tenant assignments are automatically revoked as soon as the trust relation is removed. Depending upon implementation, the assignment issuer may also choose to manually clear or even keep the nonfunctional hanging cross-tenant assignments for future use.

(P), and *tenants* (T). The definition of T and P are identical to those in CTTM while the definition of U is changed. Users are no longer owned by tenants but the roles are, while users are members of roles. RO depicts the many-to-one ownership relation from R to T.

USERS. A user is a global identity of an individual or a process. It is authenticated as a federated ID [10] which is globally unique for all the tenants in the cloud service. A user can be assigned to multiple roles in multiple tenants. In the car rental example, Bob is a user with the student role in $UTSA$. At the same time, he could also be a member of the customer role in $AVIS$. In this way, having different roles in different tenants does not change the user identity.

ROLES. A role is a job function associated with a tenant. A role belongs to a single tenant while a tenant may own multiple roles. Basically, a role is a pair of role name and tenant and is written in the format of "$role\_name\#tenant$". Sticking with the car rental example, the student role in $UTSA$ is noted as $student\#UTSA$ which is not associated with any tenant other than $UTSA$.

The formal definition of RB-CTTM follows.

Definition 2 *The role-based cross-tenant trust model (RB-CTTM) has the following components.*

- *T, P, TT and PO are unchanged from CTTM; U and R are finite sets of global users and roles respectively;*

- *$RO \subseteq R \times T$, is a many-to-one relation mapping each role to its owner tenant; correspondingly, $roleOwner(r : R) \rightarrow T$, is a function mapping a role to its owner tenant where $roleOwner(u) = t$ iff $(r, t) \in RO$;*

- *$UA \subseteq U \times R$, is a many-to-many user-to-role assignment relation;*

- *$PA \subseteq P \times R$, is a many-to-many permission-to-role assignment relation requiring that $(p, r) \in PA$ only if*
  $permOwner(p) \equiv roleOwner(r) \vee$
  $permOwner(p) \trianglelefteq_\alpha roleOwner(r) \vee$
  $roleOwner(r) \trianglelefteq_\beta permOwner(p) \vee$
  $permOwner(p) \trianglelefteq_\gamma roleOwner(r)$,
  where only one of the $\trianglelefteq$ requirements can apply depending on the nature of TT;

- *$RH \subseteq R \times R$ is a partial order on R called role hierarchy or role dominance relation, also written as "$\geq$", requiring that $r_2 \geq r_1$ only if*
  $roleOwner(r_1) \equiv roleOwner(r_2) \vee$
  $roleOwner(r_1) \trianglelefteq_\alpha roleOwner(r_2) \vee$
  $roleOwner(r_2) \trianglelefteq_\beta roleOwner(r_1) \vee$
  $roleOwner(r_1) \trianglelefteq_\gamma roleOwner(r_2)$,
  where only one of the $\trianglelefteq$ requirements can apply depending on the nature of TT.

Note that in order to enable role activation, a session entity component and corresponding functions could also be added to RB-CTTM like those in RBAC. However, we do not discuss sessions in this paper since they are not critical to the focus of this paper.

Since the users are global in RB-CTTM, UA is an arbitrary relation without limitation of tenants, unlike RH and PA. Both RH and PA are tenant-aware assignments with can be intra-tenant or cross-tenant. Intra-tenant RH and PA are similar to those with the same names respectively in RBAC models [16]. Each cross-tenant assignment requires at least one appropriate trust relation as specified in Definition 2.

With RH senior roles inherit permissions from their junior roles so that the permissions are transitively inherited to the users. We can authorize a cross-tenant access from a role to a permission in different tenants through either RH or PA. But, it is worth noting that in a Type-γ trust relation, cross-tenant PA is not recommended to be allowed. If the trustor (the permission owner) allows the trustee (the role owner) to make cross-tenant PA, then the trustor will never know which trustee's roles inherit the permissions, let alone controlling the inheritance. A better practice could be making only intra-tenant PA and cross-tenant RH. In this way, the trustor can at least control the inheritance of its own permissions by PA.

3.4 Feasibility in the Cloud

According to the essential characteristics of the cloud discussed in Section 2.2, a suitable platform should be developed to better enforce our cross-tenant access control models in cloud computing. We propose a cloud-based platform named multi-tenant authorization as a service (MTAaaS). Its architecture is shown in Figure 4.

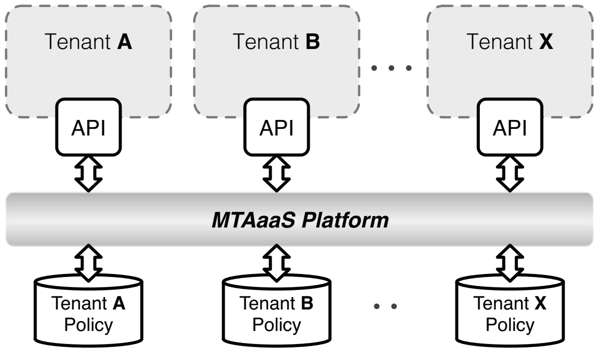

Figure 4. MTAaaS Architecture

MTAaaS serves in the enforcement layer of the PEI framework for application-centric security [26]. It is basically a middleware centrally controlling multi-tenant accesses in the cloud service. A user in a tenant can submit cross-tenant or intra-tenant access requests to the MTAaaS platform through standardized MTAaaS API. Upon receipt of the requests, the MTAaaS platform will evaluate the requests referring the policy of the tenant. The evaluation results will then be sent back to the tenants and enforced by the MTAaaS API. Each tenant has its own access control policy stored in the cloud service and managed through the MTAaaS platform. These policies are defined according to one or more of the multi-tenant access control models [11, 28, 29], including CTTM and RB-CTTM. We believe the MTAaaS platform is consistent with the essential characteristics of the cloud and capable to manage and enforce cross-tenant access control policies.

4. Related Work

Cross-tenant access control in the cloud can be aligned with the cross-domain access control mechanisms in traditional environments. RBAC [16, 27] enables fine-grained access control in a single domain. In order to enable multi-domain access control and inherit the benefit from RBAC, many extensions of RBAC have been proposed [15, 20, 21, 31]. These approaches rely on a centralized authority to define and maintain cross-domain policies. However, in a cloud environment, different tenants are usually from totally different organizations with independent authority. It is highly unlikely they will agree to have a central party to manage access control for them. Therefore, the approaches using centralized authority are not suitable for the cloud.

Another approach extending RBAC to achieve cross-domain access control is using delegation [4, 7, 18, 30]. In these approaches, users may delegate their entire or partial permissions to others. The delegation relations are managed by users in a decentralized manner. But in the cloud, users and tenants are dynamically and rapidly changing. The delegation rules managed by users cannot support the agility of cross-tenant access needs.

In traditional distributed environments, federated identity and authorization services are utilized to manage cross-domain access control. Federated identity [10] enables authenticating strangers by sharing identity information among federated parties who trust each other equally in a bilateral way which is ineffective in the cloud as we discussed before. Moreover, it is overwhelming to maintain the federation when the tenants keep changing. Authorization services [5, 8, 12, 22, 25] are developed to control collaborative resource sharing among different *Virtual Organizations* (VOs) in grids using credentials. Due to the differences between the grid and the cloud [17], the maintenance of cryptographic credentials is very costly in cloud settings. Moreover, the centralized facility of clouds provide opportunities for policy-driven authorization services. Therefore, to build collaborations among tenants, such credential-driven approaches are neither appropriate nor necessary.

In cloud environments, cross-tenant access control mechanisms are still under discussion while role-based trust models [11, 19, 28, 29] are believed to be beneficial. Our work in this paper is exploring towards a general model (CTTM) which consolidates the trust models in these approaches. The types of cross-tenant trust relations in CTTM map to each of the existing approaches. Role-based Trust-management (RT) framework [19] implements Type-α trust with cryptographic credentials. Both of Multi-Tenancy Authorization System (MTAS) [11, 29] and Multi-Tenant RBAC (MT-RBAC) [28] are designed for multi-tenant cloud environments and map respectively to Type-β and Type-γ trust relations.

5. Conclusion and Discussion

In summary, we argue proper trust relations are beneficial in facilitating cross-tenant access control in cloud computing. Through a systematic analysis of cross-tenant trust relations, we propose a formalized cross-tenant trust model (CTTM) and its role-based extension (RB-CTTM). Moreover, we propose a multi-tenant authorization as a service (MTAaaS) platform for enforcing our models in the cloud.

In the future, we plan to investigate attribute-based extensions of CTTM and other possible models which are compatible with our MTAaaS platform. Also, we are working on the implementation of the MTAaaS platform in OpenStack [1], an open source cloud management system. The implementation will encompass cross-tenant access control mechanisms.

Acknowledgement

This work is partially supported by grants from the National Science Foundation and AFOSR MURI program.

References

[1] Openstack: Open source software for building private and public clouds. http://www.openstack.org.

[2] Trust (social sciences). http://en.wikipedia.org/wiki/Trust_(social_sciences).

[3] Single Sign-On with SAML on force.com. http://wiki.developerforce.com/page/Single_Sign-On_with_SAML_on_Force.com, 2013.

[4] M. Alam, X. Zhang, K. Khan, and G. Ali. xDAuth: A scalable and lightweight framework for cross domain access control and delegation. In *Proceedings of the 16th ACM Symposium on Access Control Models And Technologies (SACMAT)*, pages 31–40. ACM, 2011.

[5] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lőrentey, and F. Spataro. From gridmap-file to VOMS: managing authorization in a grid environment. *Future Generation Computer Systems*, 21(4):549–558, 2005.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical report, EECS Department, University of California, Berkeley, Feb 2009.

[7] L. Bauer, L. Jia, M. K. Reiter, and D. Swasey. xDomain: cross-border proofs of access. In *Proceedings of the 14th ACM Symposium on Access Control Models And Technologies (SACMAT)*, pages 43–52. ACM, 2009.

[8] E. Bertino, P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Ferrari. Towards supporting fine-grained access control for grid resources. In *Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS)*, pages 59–65. IEEE, 2004.

[9] R. Bhatti, E. Bertino, and A. Ghafoor. X-FEDERATE: A policy engineering framework for federated access management. *IEEE TSE*, 32:330–346, 2006.

[10] R. Bhatti, E. Bertino, and A. Ghafoor. An integrated approach to federated identity and privilege management in open systems. *Communications of the ACM*, 50(2):81–87, 2007.

[11] J. M. A. Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray. Toward a multi-tenancy authorization system for cloud services. *IEEE Security and Privacy*, 8(6):48–55, Nov/Dec 2010.

[12] D. W. Chadwick and A. Otenko. The permis x. 509 role based privilege management infrastructure. volume 19, pages 277–289. Elsevier, 2003.

[13] F. Chong, G. Carraro, and R. Wolter. Multi-tenant data architecture. http://msdn.microsoft.com/en-us/library/aa479086.aspx, 2006.

[14] R. F. Chong. Designing a database for multi-tenancy on the cloud. http://www.ibm.com/developerworks/data/library/techarticle/m-1201dbdesigncloud/index.html, 2012.

[15] E. Cohen, R. K. Thomas, W. Winsborough, and D. Shands. Models for coalition-based access control (CBAC). In *Proceedings of the 7th ACM Symposium on Access Control Models And Technologies (SACMAT)*, pages 97–106. ACM, 2002.

[16] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. on Information and System Security (TISSEC)*, 4(3):224–274, Aug. 2001.

[17] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop (GCE)*, pages 1–10. IEEE, 2008.

[18] E. Freudenthal, T. Pesin, L. Port, E. Keenan, and V. Karamcheti. dRBAC: distributed role-based access control for dynamic coalition environments. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, pages 411–420. IEEE, 2002.

[19] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE, 2002.

[20] 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.

[21] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo. Policy decomposition for collaborative access control. In *Proceedings of the 13th ACM Symposium on Access Control Models And Technologies (SACMAT)*, pages 103–112. ACM, 2008.

[22] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino. Efficient integration of fine-grained access control and resource brokering in grid. *The Journal of Supercomputing*, 49(1):108–126, 2009.

[23] J. McKenty. Nebula's implementation of role based access control (RBAC). http://nebula.nasa.gov/blog/2010/06/03/nebulas-implementation-role-based-access-control-rbac/, 2010.

[24] P. Mell and T. Grance. The NIST definition of cloud computing. Special Publication 800-145, 2011.

[25] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks*, pages 50–59, 2002.

[26] R. Sandhu. The PEI framework for application-centric security. In *Proceedings of the 5th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 1–6, 2009.

[27] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[28] B. Tang, Q. Li, and R. Sandhu. A multi-tenant RBAC model for collaborative cloud services. In *Proceedings of the 11th IEEE Conference on Privacy, Security and Trust (PST)*, 2013.

[29] B. Tang, R. Sandhu, and Q. Li. Multi-tenancy authorization models for collaborative cloud services. In *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems (CTS)*, 2013.

[30] X. Zhang, S. Oh, and R. S. Sandhu. PBDM: a flexible delegation model in RBAC. In *Proceedings of the 8th ACM Symposium on Access Control Models And Technologies (SACMAT)*, pages 149–157. ACM, 2003.

[31] Z. Zhang, X. Zhang, and R. Sandhu. ROBAC: Scalable role and organization based access control models. In *Proceedings of the 1st IEEE International Workshop on Trusted Collaboration (TrustCol)*, pages 1–9, 2006.