Authorization Federation in IaaS Multi Cloud

Navid Pustchi
Institute for Cyber Security
Department of Computer Science
Univ of Texas at San Antonio
tam498@my.utsa.edu

Ram Krishnan
Institute for Cyber Security
Department of Electrical and Computer Engineering
Univ of Texas at San Antonio
ram.krishnan@utsa.edu

Ravi Sandhu
Institute for Cyber Security
Department of Computer Science
Univ of Texas at San Antonio
ravi.sandhu@utsa.edu

ABSTRACT

As more and more organizations move to cloud, it is inevitable that cross-organizational collaboration will need to be supported in the cloud. In this paper, we explore models for collaboration among clouds whose resources are distributed across multiple cloud service providers. In particular, we focus on collaboration and sharing of resources in an infrastructure as a service cloud, where compute resources complemented with storage and networking are offered as a service to customers by cloud service providers. We develop a multi-cloud trust model at different scopes of administration. At cloud level, infrastructure resources can be shared between clouds. At lower administrative scopes, cloud service providers are able to share their resources and service instances among customers within multiple clouds. Finally, we formally specify the administrative aspects of multi-cloud collaboration models we develop. We have implemented a proof of concept prototype based on the administrative model of OpenStack, the *de facto* open-source software for building infrastructure as a service clouds.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Data sharing*

Keywords

Cloud Computing; Multi Cloud; Authorization Federation; Cloud Federation; Distributed Access Control; Security; Trust Management.

1. INTRODUCTION

Cloud computing is revolutionizing the way organizations avail IT resources. Its service models and on-demand features have been embraced by cloud service consumers and motivated enterprises to move towards integrating their computing resources into cloud. Cloud computing pay-per-use business model can lower costs for organizations, and its agile infrastructure leads to modulated investment for businesses. Elasticity and low-maintenance cost makes it attractive for organizations to deploy their resources in cloud.

There are two main scenarios that arise for multi-cloud collaboration. Firstly, as organizations move more and more of their IT resources to cloud, it is likely that collaboration activities with other organizations will need to occur in the cloud platform as well, since many resources of organizations will be hosted in the cloud. Secondly, a relatively large organization might utilize multiple cloud service providers for reasons including availability and reliability, and hence will need the ability to consolidate resources across multiple cloud service providers for day to day operations. This situation may also arise due to merger and acquisitions where the underlying companies are deployed on different cloud providers.

In order to make use of services from multiple clouds a reality, several technical barriers need to be resolved. The research community is beginning to establish architectures and standards for collaboration across multiple clouds [3, 18, 19, 20]. Collaboration activity in multi-cloud is primarily concerned about what operations that users who belong to a customer in one cloud can perform on resources owned by a customer in another cloud. Thus a critical challenge in facilitating multi-cloud collaboration is to allow customers across multiple clouds to precisely control what resources they are willing to share with other customers, and what operations are authorized on those shared resources.

In this paper we focus on trust models for collaboration across multiple infrastructure as a service (IaaS) cloud service providers [13]. We assume cloud collaborations we are dealing with are all on homogeneous platforms, i.e., running the same cloud IaaS system. This allows us to focus more on the trust models for multiple cloud, and bypass integration issues regarding operations of different clouds. To be concrete we utilize OpenStack [16], the open-source cloud platform for IaaS, for our discussion.

To motivate the problem, we use an example illustrated in Figure 1, in which an inter-university research community is formed. We use The European Organization for Nuclear Research (CERN) example where the amount of data captured currently is 110 petabytes and 50 petabytes are added each year. The amount of data stored in participating institutes is so large that transmitting data to perform analysis is not

SCC'15, April 14-17, 2015, Singapore, Singapore

http://dx.doi.org/10.1145/2732516.2732523

practical. Moreover adding accounts for all the participating institutes' users in each individual cloud is also impractical.

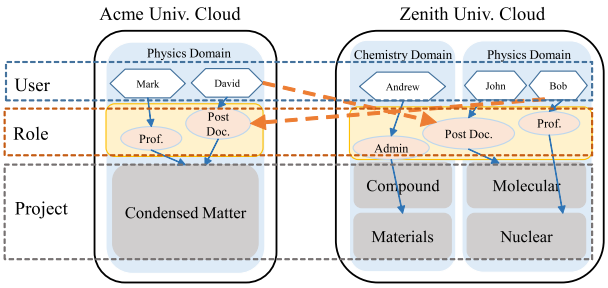


Figure 1: A Multi-Cloud Collaboration Example

A natural way to solve the issue is to get institutes collaborating together to form an inter-university research community called CERN. We have two CERN member universities Acme and Zenith running OpenStack as their cloud platform. Bob is a professor in physics domain in Zenith. For Bob to properly perform his analysis he should have access to Acme Cloud's project Condensed Matter. There should be cross-cloud access which enables Bob to perform his analysis. This can allow Bob to create a virtual machine (VM) in Acme cloud's Condense Matter project and perform analysis. Meanwhile David a postdoc in Acme Cloud requests to access Molecular domain in Zenith Cloud which he normally cannot access with current specifications of cloud systems such as OpenStack. This example is a typical use case for collaboration among multiple cloud providers. There are, similar use cases such as an organization which has resources distributed across multiple cloud service providers for certain security reasons and wishes to merge the administrative controls over all resources while each cloud still has separate administration. By enabling cross-cloud access we achieve the following benefits.

- We eliminate the need to provision users in every collaborating organization.
- Inter-cloud and intra-cloud assignments are differentiated and administered separately.
- Each participating organization has some degree of control over organizations' relationship.

The remainder of this paper is organized as follows. Section 2 characterizes federation, cloud federation, and multi-cloud, and defines a trust framework and scope of collaboration. Accordingly multi-cloud trust model and formalization of administrative models are presented. In Section 3 implementation of proposed model is discussed. Section 4 discusses related work and section 5 gives our conclusions.

2. MULTI CLOUD COLLABORATION

In real life, collaboration among organizations is inevitable due to growing challenges of global competition, rapid changes and increasing complexity of organizational structures. Organizations should be able to quickly come together and collaborate to solve a specific problem or exploit a specific opportunity. Such a group of collaborative organizations forms a federation. A *federation* can be defined as an organizational structure where multiple organizations have set up collaborative agreements [6]. Each organization has a separate administration and domain bounded to other organizations by trust agreements. The concept of federation has a long and varied history in computer systems. Just as one example, the notion of virtual organizations [15] has been developed in the distributed systems and grid computing communities going back almost two decades. It is beyond the scope of this paper to give a comprehensive review of federation in computer systems.

Our focus in this paper is on cloud federation. For simplicity we will henceforth understand the term federation to mean cloud federation. Cloud federation is a multi-faceted concept and has been treated in different ways in the literature. A cloud federation can be defined as a collaboration of cloud service providers and identity providers in order to share their services and resources within participating clouds based on trust agreements. In the following we characterize cloud federation which is compliant with NIST definition of cloud computing [14]. We can distinguish cloud federations based on type of *service, platform, trust,* and *coupling* offered in a federation. For our purpose in this paper, cloud federation can be described by specific characteristics in terms of service, platform, trust and coupling of services, as shown in Figure 2. The indicated red path is the one we are particularly focussed on in this paper.

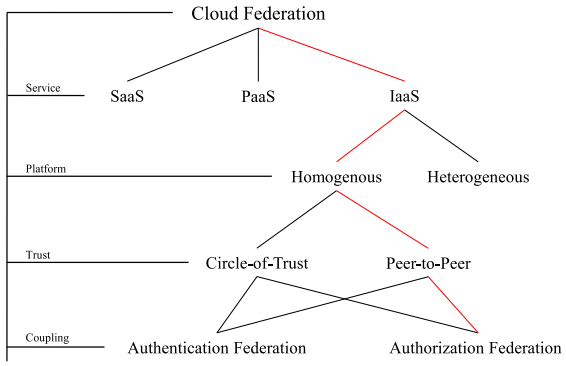


Figure 2: Characteristics of Cloud Federation

Service (IaaS vs PaaS vs SaaS). Cloud federation service models illustrate the homogeneity or heterogeneity of services offered to federated cloud participants. In IaaS, services are generally homogeneous because services are generally computation, storage, networks, etc. Whereas, in PaaS and SaaS the services offered can be heterogeneous as well. For example, Google account (OpenID 2.0) is a heterogeneous federation within the Google organization for different services from email and storage to mobile payment.

Platform (Heterogeneous vs Homogeneous). Cloud federation deployment models such as Private, Public and Hybrid clouds, based on their platform can form homogeneous or heterogeneous cloud federations. For example, a federation of an OpenStack Private cloud and an OpenStack Public cloud would form a homogeneous multicloud. On the other federation of an OpenStack Private cloud with a pro-

prietary Public cloud such as AWS or Azure would form a heterogeneous multicould.

Trust (Circle-of-Trust vs Peer-to-Peer). In a cloud federation trust relations among federating cloud members defines the type of collaboration that is enabled. In Circle-of-Trust type of federation, a group of clouds shares specific resources where trust relationships are usually established by a set of contracts defining the obligations and rights each party has. Adding additional clouds requires all cloud federation members to agree on trusting the new member [11]. CERN is such a cloud federation in which institutions joining its Circle-of-Trust can gain certain access to analytic data and computation resources across its cloud federation members. In a Peer-to-Peer trust agreement, the trust is established between each two members. In the commercial setting, federating clouds with peer-to-peer trust is more appropriate due to limited trust, and presumably enhanced security. Identity federation proposed by Chadwick et al [5] in OpenStack Icehouse [17] is such an example, wherein an identity provider can federate its users to OpenStack. Keystone (OpenStack identity service) to Keystone federation in the OpenStack Juno release enables two OpenStack clouds to form a federation with Peer-to-Peer trust.

Coupling (Authentication vs Authorization). In a cloud federation users from a cloud should be able to access services from trusted clouds. Authentication federation and authorization federation are concerned with such mechanisms. Authentication federation is concerned with mechanisms to authenticate users in clouds other than their home cloud (where they are initially authenticated). Authorization federation, on the other hand, is concerned with mechanisms to determine which authenticated users from trusted clouds have access to which resources in federated service providers. It is necessary for both participating clouds to have some degree of control over the authorization of federated user's access to shared resources. We further discuss these concepts in the following sections.

Cloud federation is a broad term which includes federations of clouds and identity providers. Multi-cloud has been used in the literature [20] extensively as a federation term as well. For our purpose multi-cloud is a cloud federation where all collaborating members are exclusively clouds. In particular, we do not consider a cloud and an external identity provider to constitute a multi-cloud. Specifically, we define *multi-cloud* as a collaboration of multiple cloud service providers (Private or Public) within different administrative domains to provide integrated services at different service models (Infrastructure, Platform and Software). In this paper, we focus on authorization federation in homogeneous multi-cloud systems in Infrastructure as a Service as highlighted in red in Figure 2. Whether a multi-cloud is a Circle-of-Trust or Peer-to-Peer, it is important to characterize the trust relations among clouds in a multi-cloud environment. Figure 3, characterizes potential trust relations between two clouds in a multi-cloud. In this paper, we identify what a simple yet very useful Peer-to-Peer trust relation means in cross-cloud authorization, and how the trust relation interacts with the existing intra-cloud access control model.

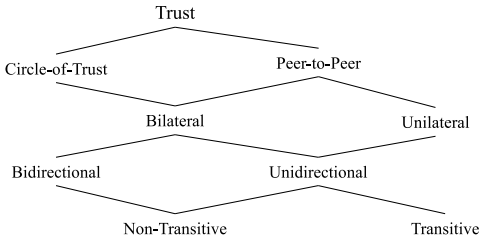


Figure 3: Trust Characteristics

When a trust relation is initiated, if both trustor and trustee agreement on trust establishment is required, then the trust is regarded as *bilateral*, otherwise it is *unilateral*. In Peer-to-Peer trust both initiation types are possible, but in a Circle-of-Trust, the initiation should be confirmed by all federation members. Consequently, it is a bilateral (actually multilateral) trust relation. In a trust relation, when both participating clouds are equally enabled by the trust, it is considered *bidirectional*. Conversely, in a *unidirectional* trust, the actions are available only on one side or the other. Unilateral trust should typically leads to a unidirectional trust, whereas bilateral trust can be bidirectional or unidirectional. In a multi-cloud federation such as A, B and C, if A trusts B and B trusts C implies A trusts C, then trust is said to be *transitive*. Otherwise, the trust is *non-transitive*. In this paper, trust relations in multi-clouds are specified as Peer-to-Peer, unilateral, unidirectional and non-transitive. Moreover, they are *reflexive* meaning each cloud trusts itself.

In this section, we discus authentication federation in 2.1. In section 2.2, authorization federation within a homogeneous multi-cloud system, trust types and administrative boundaries are discussed. We propose our multi-cloud trust model in section 2.3. The administrative model formalization is presented in section 2.4.

2.1 Authentication Federation

The basic concept of authentication federation is trusted relationship between identity providers and service providers (who themselves can also be identity providers). Federation Identity Management has been widely researched, providing solutions by enabling propagation of identity information to services located in different administrative domains [4, 5].

For our purpose, we define *identity federation* in multi-cloud as authenticating users in a cloud service provider other than their registered identity provider based on existing trust between the two parties. Several frameworks have been developed such as SAML 2.0 [10], Liberty Identity Web Services Framework (ID-WSF) 2.0 [23], and WS-Federation [1] to enable authentication federation. Currently, proposed solutions for federation in cloud platforms such as OpenStack federation, use SAML for exchanging authentication data between federated parties.

2.2 Authorization Federation

Authentication data in authentication federation will generally include some authorization information. In practice, however, the two parties typically have to manually pro-

gram the translation of authorization data from one cloud to the other, which is labor intensive and error-prone. This requires special privileges in the identity accepting cloud, contrary to the self-service spirit of cloud computing.

It is clear that a provider wants to control access to its resources in a multi-cloud system. Mechanisms to deal with access rights of identities, must enable mapping and administration of access control policies from user's cloud to service providers within existing trust without requiring special provisioning. In other words multi-cloud sharing mechanisms must retain the essential self-service nature of cloud services.

We define *authorization federation* as assigning authenticated users to resources based on trust relations in a multi-cloud federation. Authorization federation in multi-cloud must be *adoptable* (with minimal modifications by current cloud platforms), *decentralized*, *scalable* (each cloud may have multiple trust relations), *dynamic* (federation can be established and ended at any time), and *reflective* (any administrative changes in user-role assignments locally reflects related cross-cloud assignments).

Currently, authorization protocols such as OAuth [9] enables a simple way to verify the access level of a request for a web service. Such authorization protocols provide a mechanism for application users to delegate access to a third-party to work on behalf of the user (within authorization server and token delegation). Such mechanisms could be suitable in authorization federation within SaaS service models. However, they are not suitable for IaaS service models. Recently, OpenStack Keystone's federation extension for Juno release, introduced a SAML generator to map token permissions to SAML assertions and a mapping engine to map these assertions to groups (in OpenStack a group is simply a collection of users) and roles on service provider side. This enables each cloud administrator to map local users and groups to trusted cloud's users and groups manually for each trust relation in Peer-to-Peer multi-cloud federation within OpenStack clouds. While these efforts create a basic OpenStack to OpenStack federation, they lack the characteristics of a multi-cloud authorization federation outlined above. In order to develop such a fine grained authorization federation, it is necessary to define trust relations and specify scope of cross-cloud access within homogeneous multi-cloud environment.

2.2.1 The Concept of Trust

Trust determines how clouds interact with each other, including which and how much information they share in a trust relationship. Based on such trust properties, in the following we identify four potential types of trust relations to establish and control cross-cloud access in multi-cloud federation. Of these types α, β and γ are adapted from similarly defined trust types in intra-cloud systems [21], whereas type δ is newly introduced.

In such trust relations who controls the trust relation's existence and who controls the authority to issue cross-cloud assignments determines the type of trust. In the following, we use two clouds A and B where each has a set of users and resources and cross-cloud assignments are $users \rightarrow resources$.

We use "$\trianglelefteq$" as a trust relation notion where $A \trianglelefteq B$, states that A trusts B.

Type-α. Trustor grants inter-cloud access to trustee. It is perhaps the most intuitive trust meaning. By trusting a cloud, trustor shares certain resources with trusted cloud. *If $A \trianglelefteq_\alpha B$, cloud A is authorized to assign B's users to cloud A's resources. In such trust type, A controls trust relation existence and cross-cloud assignments.* Type-α trust is useful when cloud A is a resource provider and cloud B is an identity provider.

Type-β. Trustee grants inter-cloud access to trustor. *If $A \trianglelefteq_\beta B$, cloud B is authorized to assign A's users to its resources. In such trust type, A controls trust relation and B controls cross-cloud assignments.* In order for cloud A to access shared resources in cloud B, it should trust B with exposing its user set and trust B's authorization with assignments ($User_A \rightarrow Resource_B$).

Type-γ. Trustee takes inter-cloud access to trustor. *If $A \trianglelefteq_\gamma B$, cloud B is authorized to assign its users to cloud A's resources. In such trust type, A controls trust relation and B controls cross-cloud assignments.* Cloud A exposes its selected resources to share with trusted cloud B ($User_B \rightarrow Resource_A$).

Type-δ. Trustee controls intra-cloud access within trustor. *If $A \trianglelefteq_\delta B$, cloud B is authorized to assign A's users to A's resources. In such trust type, A controls trust relation and B controls intra-cloud assignments within A.* Cloud A exposes part or entire set of users and resources to cloud B ($User_A \rightarrow Resource_A$). This trust type is necessary for delegating administration across two clouds.

2.2.2 The Administrative Realms of Collaboration

When dealing with the concept of trust in a multi-cloud environment, we can identify how types of trust define authority over trust relations and assignments. Similarly, in such a federation, the scope of authorization federation within a trust relation defines type of federated resources being shared such as services, resource containers, operations, or data objects. In order to characterize scope of trust in authorization federation, we identify three administrative realms: Cloud, Domain (Tenant), and Project within a cloud system. In Figure 4, the conceptual administrative boundary of each realm with relation to services and users in a cloud is illustrated. Such conceptual structure of entities presents an extension the OSAC model for OpenStack [22].

In an IaaS cloud system, *Cloud* realm holds authority over services such as compute, storage, network, and identity, as well as over lower administrative realms such as domains and projects. If scope of trust between two clouds is cloud realms, then such services can be shared depending upon the trust type. Such collaboration is helpful for load balancing within a multi-cloud federation. A *Domain* is an administrative realm of users, groups and projects. If administrative scope of trust is Domain, users and projects within two clouds can be shared by cross-domain assignments. It is useful to enable access within a multi-cloud federation by enabling cross-cloud assignment for federated users and resources. A *Project* administrative realm is a container for

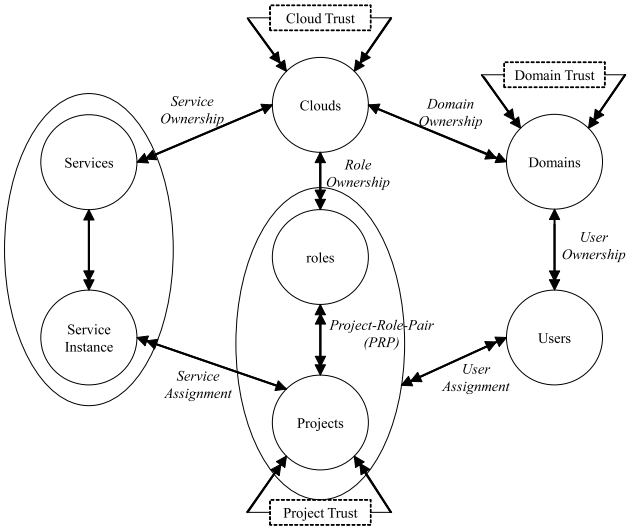


Figure 4: Administrative Realms in a Cloud

cloud resources. Each project owns its service instances (operations and data objects instantiated from services), such as VMs which are an instance of compute service. While services are within cloud realm, such as compute service which is responsible for VM provisioning, a project manages multiple service instances and each services segregates its resources across multiple projects. Such scope of trust is desirable when the objective of collaboration is enabling cross-cloud assignment for a specific service instance. For example, within a trust between two clouds, a cloud wishes to only share online sales project from sales domain.

Based on the scopes of trust characterized above, each cloud's administrative realm active in a trust relation can share its specific type of resources within the trust relationship to another cloud in a multi-cloud federation.

2.3 Multi Cloud Trust

In the following we discuss some use cases for trust models at different realm granularity to enable inter-cloud access in a multi-cloud federation.

2.3.1 Cross Cloud Trust

It is natural for clouds to share their infrastructure resources. Sharing infrastructure is useful in case of handling large bursts of traffic, load balancing, outsourcing, etc. In order to enable clouds to share their physical infrastructure, it is necessary to create a controlled trust relation within *cloud* administration realm of participating clouds. In type-β by trusting a cloud, trustor agrees that trustee allocates its services to trustee's infrastructure. In Figure 5, such a β cross-cloud trust is depicted. It illustrates $cloud_A \trianglelefteq_\beta cloud_B$ meaning $cloud_B$ is authorized to assign VMs in $cloud_A$ to its physical shared resources while $cloud_A$ can initiate and end this collaboration. In such a collaboration, in order for clouds to have access within shared infrastructure they should agree on granting trustee cloud, assignment authorization and visibility to VMs they wish to federate. Such trust enables $cloud_A$, when there is shortage of available physical resources to still provision users with VMs in trusted $cloud_B$. Figure 6, depicts sequences to establish, assign and remove resources within $cloud_A \trianglelefteq_\beta cloud_B$ between two cloud service providers.

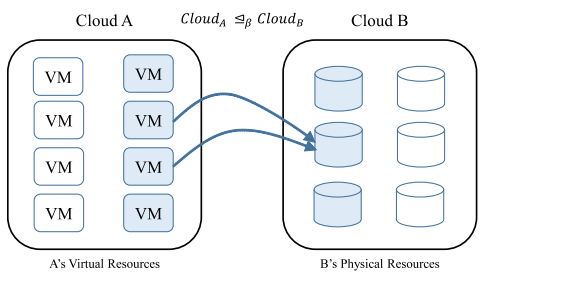


Figure 5: Cross-Cloud Resource Allocation in Two Federated Clouds

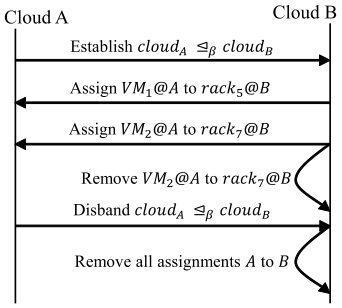


Figure 6: Cross-Cloud Trust Type-β Enforcement Sequence

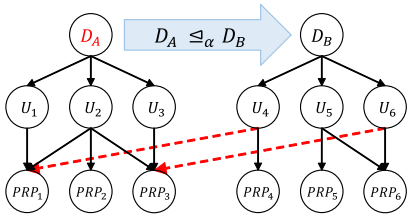
In a multi-cloud federation when a cloud is willing to share its resources by delegating access control to trusted cloud, type-γ is useful. In type-γ by trusting a cloud, trustor shares its physical resources with trusted clouds and authorize trusted clouds to assign their VMs to its resources.

2.3.2 Cross Domain Trust

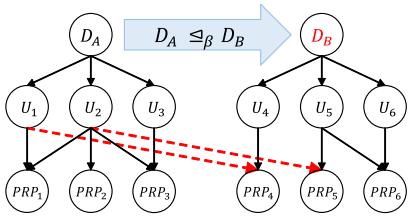
In an organization which owns multiple clouds across independent cloud service providers or a collaboration group with many member institutes, enabling users to access resources across clouds is desirable. In our multi-cloud trust model, we enable user assignment to shared projects between two clouds upon the trust relation among trusted domains. As defined in section 2.2.2 each *domain* realm has its set of users and projects. Users are assigned roles with respect to projects or project-role-pairs (PRPs).

We apply four trust types $(\alpha, \beta, \gamma, \delta)$ to authorize cross-domain assignments within homogeneous multi-cloud collaboration. Type-α is illustrated in Figure 7a. It enables user assignments between an identity provider's users and cloud's PRPs. This intuitive trust type enables a domain such as $domain_A$ to share its PRPs by trusting an IdP such as IdP_B ($domain_A \trianglelefteq_\alpha IdP_B$). $domain_A$ is authorized to establish trust relation and control user assignments from B to its PRPs. Type-β is practical for sharing resources while privacy of resources is a concern in collaboration. In Figure 7b, $domain_A \trianglelefteq_\beta domain_B$ states that $domain_A$ agrees to grant user visibility (users to be shared) to $domain_B$. In such a trust relation $domain_B$ administers cross-domain user assignments. Type-γ is useful to share projects within a

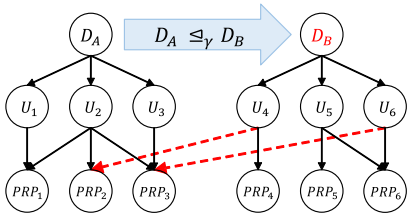
collaboration group of clouds. Figure 7c depicts that trustee $domain_B$ controls user assignments to $domain_A$'s PRPs. Type-δ in Figure 7d enables trusted cloud ($domain_B$) to administer intra-cloud assignments in trustor ($domain_A$). Such trust type is useful to achieve administration federation in a multi-cloud environment.



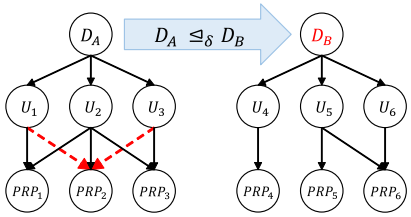
(a) A trusts B in Type-α



(b) A trusts B in Type-β



(c) A trusts B in Type-γ



(d) A trusts B in Type-δ

Figure 7: Cross-Domain Trust User Assignments

2.3.3 *Cross Project Trust*

Trust between clouds' *projects* in a multi-cloud federation authorizes sharing of their service instances. Trust relations α, β, γ, and δ are applicable to such administration realm similar to user assignments in section 2.3.2. In such trust relations users are assigned to projects' service instances such as VMs or object storages within a project trust scope. For example, when sales project wishes to share its sales VMs or sales databases cross-project trust is sufficient and there is no need to establish a domain trust between collaborating clouds.

2.4 Administrative Model

In this section, we formalize the multi-cloud trust model presented in section 2.3. Due to space limitations, we formally specify a set of administrative operations necessary for Type-β cross-cloud trust in table 1 (administrative models for type α, γ, and δ is provided in appendix A). Semantics for cloud and domains to establish Type-β trust and corresponding cross-domain user assignments are as follows.

Establish$_\beta$: An administrator (cloud or domain) user establish trust to another domain in trusted cloud. In table 1 U is global set of users and D set of domains. In column 2, for Establish_trust to succeed, the trustor user must have admin role in its domain. Subsequently, in column 3, domain trust set DT_β is updated with collaborating domains.
Assignment$_\beta$: A domain admin user u_1 is allowed to assign trustor domain's user u_2 to its PRP (p_1, r_1). Subsequently, user assignment set UA in trustor domain is updated to reflect that trustor user u_2 is assigned to trustee PRP (p_1, r_1). In table 1 P is set of projects and R is set of roles in each cloud.
Unassignment$_\beta$: Unassignment operation is similar to assignment, except assignment by user admin u_1 should be removed from user assignment set UA in trustor domain. To this end, we update UA by removing (u_2, p_1, r_1) from UA.
Disband$_\beta$: The semantics for this operation is to end collaboration between two domains. To this end, first we remove all assignments from trustee domain admin u_1 in trustor's domain UA set, then we remove trust relation $(user\_owner(u_1), d_1)$ between two domain from domain trust set DT.

3. IMPLEMENTATION

Currently, OpenStack is *de facto* open-source software to deploy IaaS platforms. In order to explore the feasibility of our multi-cloud trust model, we implemented a prototype in a single cloud for a cross-domain trust running Icehouse release of Keystone [17] in Devstack (OpenStack development environment). In current state Keystone although cloud admin can assign users to roles (in OpenStack roles are cloud global) within projects in other domains, it does not support any trust between domains for cross-domain access by domain admin. Our implementation should support two capabilities: (a) CRUD operations for administration of trust types between domains and (b) enabling user access to shared PRPs. We take into consideration that efficiency of

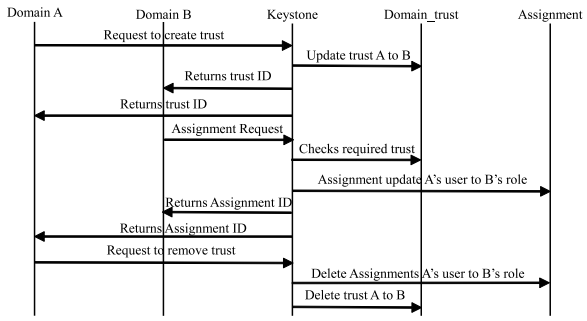


Figure 8: Cross-Cloud Domain Trust Requests Sequence

trust models enforced depends on many factors, such as the number of requests, levels of policy checks in policy engine and so on.

Implementation. The architecture of our prototype follows the Keystone design. Keystone uses MySQL database to store all user, domain, and assignments information. In

Operations	Authorization Requirements ($\rightarrow$)	Updates
$\forall u_1 \in U, \forall d_1 \in D$, **Establish**$_{\boldsymbol{\beta}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge (user\_cloud(u_1), domain\_owner(d_1)) \in CT$	$DT_\beta^{'} = DT_\beta \cup (user\_owner(u_1), d_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Assignment**$_{\boldsymbol{\beta}}$ (u_1, u_2, p_1, r_1)	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge (user\_owner(u_1) = project\_owner(p_1)) \wedge (user\_owner(u_2), user\_owner(u_1)) \in DT_\beta$	$UA_\beta^{'} = UA_\beta \cup (u_2, p_1, r_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Unassignment**$_{\boldsymbol{\beta}}$ (u_1, u_2, p_1, r_1)	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge (user\_owner(u_1) = project\_owner(p_1)) \wedge (user\_owner(u_2), user\_owner(u_1)) \in DT_\beta$	$UA_\beta^{'} = UA_\beta - (u_2, p_1, r_1)$
$\forall u_1 \in U, \forall d_1 \in D$, **Disband**$_{\boldsymbol{\beta}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge (user\_owner(u_1), d_1) \in DT_\beta$	$UA_\beta^{'} = UA_\beta - \{(u, p, r) \mid u \in U, p \in P, r \in R, (user\_owner(u), project\_owner(p)) \in DT_\beta\}, DT_\beta^{'} = DT_\beta - (user\_owner(u_1), d_1)$

Table 1: Cross-Domain Trust Type-β Administrative Model

our implementation, we added a table Domain_trust to store trust types and collaboration domains information. Also modified Assignment table in Keystone database to store required cross-domain trust information for assignments. We developed our own Keystone client for added domain administration capabilities in OpenStack and our developed modifications to Keystone. By adding trust to assignment table a domain admin with trust relation with another domain can assign users to PRPs based on trust type. In Figure 8, the sequence of administrative operations added to Keystone is shown. It depicts two collaborating domains, A and B and Keystone's inner components we added or modified for cross-domain trust. It is showing the sequence of domain A trusting domain B with type-β in our implementation. In order to establish trust among two domains, the domain administrator initiates a trust request to Keystone and based on the type of trust initiated, either trustor or trustee domain admin can establish cross-domain assignments. Since we modified user assignment process to enable cross-domain trust, there is no overhead on the operational part. When a user requests access to a service, the usual process of token and permission generation by keystone is followed as it is implemented in Icehouse. Such approach benefits the overall performance because there is no overhead on token generation, so our system performance is intact.

4. RELATED WORK

Role-Based Access Control (RBAC) [7] is the leading model for single organizations access control. Many cloud platforms such as OpenStack [16] adopted appropriate variations of RBAC for their access control mechanisms. In order to benefit RBAC capabilities across multiple organizations, model extensions such as ROBAC [25] and GB-RBAC [12] have been proposed. ROBAC manages authorization in multiple organizations which is comparable to multi-cloud, but organization collaboration is not explicitly granted in this RBAC extension. In GB-RBAC collaboration is allowed among groups, but it lacks the administration management since the administrator cannot manage users in the groups. Other RBAC extensions towards collaboration uses centralized authority to manage collaboration which is not applicable in multi-cloud scenarios. Recent work on collaboration such as CTTM [21] and OSAC-DT [22] extended RBAC to inherit its benefits toward collaboration. CTTM enables trust between tenants in a single cloud. OSAC-DT which is closely related to CTTM, extends CTTM towards compatibility with OpenStack. Our contribution is beyond domains in multi-cloud environments. Role-based delegation [2, 8, 24] models proposed to permit delegation of administration, but chained delegation relations are not dynamic and flexible enough to be deployed in multiple cloud providers since trust relations are dynamic. In multi-cloud trust model, the trust relation is initiated and controlled by trustor which makes the trust management transparent to multiple CSPs relations.

5. CONCLUSION

In this paper, we described the concept of authorization federation and presented a multi-cloud trust model to authorize collaboration at cloud, domain, and project scope of administration. We identified four types of trust applicable to administrative realms in a cloud to better facilitate cross-cloud authorization. This work was motivated in the context of homogeneous Peer-to-Peer IaaS multi-cloud federation. Further, we implemented a proof of concept prototype in Keystone Icehouse release of OpenStack. The experiments showed that the integrated cross-domain trust model is not affecting the performance.

Acknowledgement

This research is partially supported by NSF Grants CNS-1111925 and CNS-1423481.

6. REFERENCES

[1] S. Bajaj, G. Della-Libera, B. Dixon, M. Dusche, M. Hondo, M. Hur, C. Kaler, H. Lockhart, H. Maruyama, A. Nadalin, et al. Web services federation language (WS-Federation). *Retrieved April*, 14:2005, 2003.

[2] E. Barka and R. Sandhu. Framework for role-based delegation models. In *Computer Security Applications,*

2000. ACSAC'00. 16th Annual Conference, pages 168–176. IEEE, 2000.

[3] D. Bernstein and D. Vij. Intercloud security considerations. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 537–544. IEEE, 2010.

[4] D. W. Chadwick. Federated identity management. In *Foundations of Security Analysis and Design V*, pages 96–120. Springer, 2009.

[5] D. W. Chadwick, K. Siu, C. Lee, Y. Fouillat, and D. Germonville. Adding federated identity management to openstack. *Journal of Grid Computing*, 12(1):3–27, 2014.

[6] M. Decat, B. Lagaisse, D. Van Landuyt, B. Crispo, and W. Joosen. Federated authorization for software-as-a-service applications. In *On the move to meaningful internet systems: OTM 2013 Conferences*, pages 342–359. Springer, 2013.

[7] 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.

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

[9] D. Hardt. The OAuth 2.0 authorization framework. 2012.

[10] J. Hughes and E. Maler. Security Assertion Markup Language (SAML) V2. 0 Technical Overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pages 29–38, 2005.

[11] U. Kylau, I. Thomas, M. Menzel, and C. Meinel. Trust requirements in identity federation topologies. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 137–145. IEEE, 2009.

[12] 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.

[13] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[14] P. Mell and T. Grance. The NIST definition of cloud computing. 2011.

[15] Open Science Grid, Virtual Organization Summary. `http://myosg.grid.iu.edu/vosummary?all_vos=on&active=on&active_value=1&datasource=summary`.

[16] OpenStack. `http://www.openstack.org/`.

[17] OpenStack-Icehouse. `http://www.openstack.org/software/icehouse/`.

[18] M. P. Papazoglou and W.-J. van den Heuvel. Blueprinting the cloud. *IEEE Internet Computing*, 15(6):74–79, 2011.

[19] B. Rochwerger, D. Breitgand, A. Epstein, D. Hadas, I. Loy, K. Nagin, J. Tordsson, C. Ragusa, M. Villari, S. Clayman, et al. ReservoirÜwhen one cloud is not enough. *IEEE computer*, 44(3):44–51, 2011.

[20] M. Singhal, S. Chandrasekhar, T. Ge, R. S. Sandhu, R. Krishnan, G.-J. Ahn, and E. Bertino. Collaboration in multicloud computing environments: Framework and security issues. *IEEE Computer*, 46(2):76–84, 2013.

[21] B. Tang and R. Sandhu. Cross-tenant trust models in cloud computing. In *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pages 129–136. IEEE, 2013.

[22] B. Tang and R. Sandhu. Extending openstack access control with domain trust. In *Network and System Security*, pages 54–69. Springer, 2014.

[23] J. Tourzan, Y. Koga, et al. Liberty id-wsf web services framework overview. *Liberty Alliance*, 2004.

[24] X. Zhang, S. Oh, and R. Sandhu. PBDM: a flexible delegation model in RBAC. In *Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 149–157. ACM, 2003.

[25] Z. Zhang, X. Zhang, and R. Sandhu. ROBAC: Scalable role and organization based access control models. In *Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, pages 1–9. IEEE, 2006.

APPENDIX

A. ADMINISTRATIVE MODELS

In this section, we formalize multi-cloud trust types α, γ and δ presented in section 2.3. Establishing cross-domain trust relation is similar to type-β, $establish_{\boldsymbol{\beta}}$ administrative operation, hence we focus on assignment, unassignment and disband operations. In table 2, type-α operations are shown. In $assignment_{\boldsymbol{\alpha}}$, a domain or cloud admin user u_1 from trustor is authorized to assign trustee's user u_2 to it's PRP (p_1, r_1). Authorization requirements are u_1, p_1 and r_1 must belong to same trustor domain for operation to succeed. Subsequently, UA in trustee's domain is updated with (u_2, p_1, r_1) assignment. In $unassignment_{\boldsymbol{\alpha}}$, similar authorization operations to assignment is required. As a result, (u_2, p_1, r_1) is removed from trustee's UA set. In type-α to disband trust relation, initially all user-assignments $(project\_owner(p), user\_owner(u))$ which is $(trustor\_domain, trustee\_domain)$ should be removed from trustee's UA set, then domain trust relation is disbanded. Type-γ administrative operations are depicted in table 3. In $assignment_{\boldsymbol{\gamma}}$, a domain or cloud admin user u_1 from trustee is authorized to assign it's user u_2 to trustor's PRP (p_1, r_1). Authorization requirements are that u_1 and u_2 must belong to same trustee domain. Subsequently, UA in trustee's domain is updated with (u_2, p_1, r_1) assignment. $Unassignment_{\boldsymbol{\gamma}}$ operation is similar to assignment, except assignment by user admin u_1 should be removed from user assignment set UA in trustee's domain. To disband type-γ trust relation, it is mandatory to remove all user assignments $(project\_owner(p), user\_owner(u))$ from trustee's UA set before removing trust. Type-δ administrative model is shown in table 4. $Assignment_{\boldsymbol{\delta}}$ depicts an intra-domain user assignment by trustee cloud or domain admin user u_1, assigning trustor user u_2 to trustor PRP (p_1, r_1). For $assignment_{\boldsymbol{\delta}}$ and $unassignment_{\boldsymbol{\delta}}$, it is required that u_2 and p_1 domains are same trustor domain. In $disband_{\boldsymbol{\delta}}$, first we remove all intra-domain assignments in trustor domain , then we remove trust relation from DT_δ.

Operations	Authorization Requirements ($\rightarrow$)	Updates
$\forall u_1 \in U, \forall d_1 \in D$, **Establish**$_{\boldsymbol{\alpha}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_cloud(u_1), domain\_owner(d_1)) \in CT$	$DT_{\alpha}^{'} = DT_{\alpha} \cup (user\_owner(u_1), d_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Assignment**$_{\boldsymbol{\alpha}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1) = project\_owner(p_1)) \wedge$ $(user\_owner(u_1), user\_owner(u_2)) \in DT_{\alpha}$	$UA_{\alpha}^{'} = UA_{\alpha} \cup (u_2, p_1, r_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Unassignment**$_{\boldsymbol{\alpha}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1) = project\_owner(p_1)) \wedge$ $(user\_owner(u_1), user\_owner(u_2)) \in DT_{\alpha}$	$UA_{\alpha}^{'} = UA_{\alpha} - (u_2, p_1, r_1)$
$\forall u_1 \in U, \forall d_1 \in D$, **Disband**$_{\boldsymbol{\alpha}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1), d_1) \in DT_{\alpha}$	$UA_{\alpha}^{'} = UA_{\alpha} - \{(u, p, r) \mid u \in U, p \in P, r \in R, (project\_owner(p), user\_owner(u)) \in DT_{\alpha}\}$, $DT_{\alpha}^{'} = DT_{\alpha} - (user\_owner(u_1), d_1)$

Table 2: Cross-Domain Trust Type-α Administrative Model

Operations	Authorization Requirements ($\rightarrow$)	Updates
$\forall u_1 \in U, \forall d_1 \in D$, **Establish**$_{\boldsymbol{\gamma}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_cloud(u_1), domain\_owner(d_1)) \in CT$	$DT_{\gamma}^{'} = DT_{\gamma} \cup (user\_owner(u_1), d_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Assignment**$_{\boldsymbol{\gamma}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1) = user\_owner(u_2)) \wedge$ $(project\_owner(p_1), user\_owner(u_1)) \in DT_{\gamma}$	$UA_{\gamma}^{'} = UA_{\gamma} \cup (u_2, p_1, r_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Unassignment**$_{\boldsymbol{\gamma}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1) = user\_owner(u_2)) \wedge$ $(project\_owner(p_1), user\_owner(u_1)) \in DT_{\gamma}$	$UA_{\gamma}^{'} = UA_{\gamma} - (u_2, p_1, r_1)$
$\forall u_1 \in U, \forall d_1 \in D$, **Disband**$_{\boldsymbol{\gamma}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1), d_1) \in DT_{\gamma}$	$UA_{\gamma}^{'} = UA_{\gamma} - \{(u, p, r) \mid u \in U, p \in P, r \in R, (project\_owner(p), user\_owner(u)) \in DT_{\gamma}\}$, $DT_{\gamma}^{'} = DT_{\gamma} - (user\_owner(u_1), d_1)$

Table 3: Cross-Domain Trust Type-γ Administrative Model

Operations	Authorization Requirements ($\rightarrow$)	Updates
$\forall u_1 \in U, \forall d_1 \in D$, **Establish**$_{\boldsymbol{\delta}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_cloud(u_1), domain\_owner(d_1)) \in CT$	$DT_{\delta}^{'} = DT_{\delta} \cup (user\_owner(u_1), d_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Assignment**$_{\boldsymbol{\delta}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_2) = project\_owner(p_1)) \wedge$ $(user\_owner(u_1), user\_owner(u_2)) \in DT_{\delta}$	$UA_{\delta}^{'} = UA_{\delta} \cup (u_2, p_1, r_1)$
$\forall u_1, u_2 \in U, \forall p_1 \in P, \forall r_1 \in R$, **Unassignment**$_{\boldsymbol{\delta}}(u_1, u_2, p_1, r_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_2) = project\_owner(p_1)) \wedge$ $(user\_owner(u_1), user\_owner(u_2)) \in DT_{\delta}$	$UA_{\delta}^{'} = UA_{\delta} - (u_2, p_1, r_1)$
$\forall u_1 \in U, \forall d_1 \in D$, **Disband**$_{\boldsymbol{\delta}}(u_1, d_1)$	$(domain\_admin(u_1) \vee cloud\_admin(u_1)) \wedge$ $(user\_owner(u_1), d_1) \in DT_{\delta}$	$UA_{\delta}^{'} = UA_{\delta} - \{(u, p, r) \mid u, u_1 \in U, p \in P, r \in R, (user\_owner(u_1), user\_owner(u)) \in DT_{\delta}\}$, $DT_{\delta}^{'} = DT_{\delta} - (user\_owner(u_1), d_1)$

Table 4: Cross-Domain Trust Type-δ Administrative Model