MC-ABAC: An ABAC-based Model for Collaboration in Multi-Cloud Environment

Mohamed Amine Madani[1], Abdelmounaim Kerkri[2], Mohammed Aissaoui[3]
Engineering Sciences Laboratory LSI, National School of Applied Sciences
Mohammed Premier University, Oujda, Morocco[1]
Laboratory of Stochastic and Deterministic Modeling
National School of Applied Sciences, Mohammed Premier University, Oujda, Morocco[2,3]

***Abstract*—Collaborative systems allow a group of organizations to collaborate and complete shared tasks through distributed platforms. Organizations who collaborate often leverage cloud-based solutions to outsource their data and to benefit from the cloud capabilities. During such collaborations, tenants require access to and utilize resources held by other collaborating tenants, which are hosted across multiple cloud providers. Ensuring access control in a cloud-based collaborative application is a crucial problem that needs to be addressed, particularly in a multi-cloud environment. This paper presents the Multi-Cloud ABAC: MC-ABAC model, an extension of the ABAC: Attribute Based Access Control model, suitable for ensuring secure collaboration and cross-tenant access in a multi-cloud environment. The MC-ABAC introduces the notions of tenant, cloud customer and cloud service provider as fundamental entities within the model. Additionally, it incorporates multiple trust relations to enable collaboration and resource sharing among tenants in the multi-cloud environment. To demonstrate its feasibility, we have implemented the MC-ABAC model using Python technology.**

***Keywords*—ABAC model; multi-tenant; multi-cloud; collaboration; trust**

I. Introduction

Nowadays, applications and IT systems are increasingly geared towards collaboration, facilitating cooperation between organizations to attain shared objectives. These collaborative efforts optimize the utilization of distributed resources among the participating entities, leading to enhanced productivity and overall benefits. Over the years, extensive research has been conducted on the design and implementation of collaborative work environments [1], aiming to fulfill the demands of collaborative activities. In this context, collaborative applications offer innovative technologies that allow a group of users to communicate, work together, and complete shared tasks using distributed platforms.

Many organizations rely on cloud-based services provided by cloud service providers to externalize their IT infrastructure, including computing, networking, and data storage [2], [21]. This enables remote access to hardware and software over the Internet. To maintain the privacy and confidentiality of these services, the cloud service provider employs a multi-tenancy approach, segregating data and customer services into distinct tenants. Each tenant is assigned to an individual or organization utilizing the cloud service.

In multi-cloud environment, collaborating tenants may be hosted in the same cloud provider or in different cloud providers. These tenants often require access to information shared by other tenants during collaborative process. This information often contains sensitive data. Balancing collaboration and security can be challenging because collaboration aims to provide access to services and resources for those who require them, while security focuses on preserving the availability, confidentiality, and integrity of these assets and limiting access to authorized individuals only. This raises the issue of access control [25]. In order to facilitate access and collaboration across multiple tenants, a robust and fine-grained access control model is mandatory.

Traditional access control models [3], [4], [5], [6], [7] DAC, MAC, RBAC, TBAC, TMAC and others are primarily designed for defining access policies within a single organization or for specific local access control scenarios. However, they may not be adequately suitable for defining access policies in multi-tenant environments. To address the challenges of collaboration and multi-tenant access, various access control models [23], [12], [13], [14] (CTTM, MTAS, MT-RBAC, MT-ABAC, etc) have been proposed. Nevertheless, these approaches are designed for multi-tenant environment within a single cloud. They may not effectively address the access control challenges that arise in multi-cloud environments.

In this paper, we propose MC-ABAC: Multi-cloud ABAC model, as an extended version of the ABAC [8] (Attribute Based Access Control) model. MC-ABAC leverages the capabilities of ABAC [18], [24], such as flexibility and adaptability. MC-ABAC is especially designed for securing collaboration and cross-tenant access in a multi-cloud environment. It presents the concepts of tenant, cloud customer and cloud service provider as key entities in the model. Furthermore, this model introduces many trust relations in order to support resource sharing between tenants in a multi-cloud environment. Finally, we implemented this model using Python technology to demonstrate its feasibility. To the best of our knowledge, this is the first work, that aims to extend the ABAC model to enable collaboration in a multi-cloud environment.

Our main contribution in this paper is to define a multi-cloud ABAC model, with cross-tenant trust in a multi-cloud environment. This model is suitable for supporting sharing ressources between multi-cloud tenants belonging to different cloud providers. Our model takes into account the heterogeneity requirement, since cloud providers may use different and heterogeneous access control models. This is possible through the flexibility of the ABAC model, which can represent access policies defined by any model [3]. Furthermore, this model is better suited to control access to any cloud service, whether it

be IaaS, PaaS, or SaaS.

This paper is organized as follows: Section 2 presents the background of this work, focusing on a telemedicine use case and the explanation of the ABAC model. The related work is presented in Section 3. Section 4 presents the suggested MC-ABAC model. In Section 5, we introduce the trust relations. Section 6 describes the implementation architecture. Finally, we conclude in Section 7.

II. Background

The purpose of this section is to provide the necessary background information for this work. It primarily focuses on presenting a use case of telemedicine within a cloud-based collaborative environment. Additionally, it introduces the attribute-based access control model.

A. Case Study

In this study, we examine the telemedicine use case, which involves Rabat's School Hospital $SH1$ and Oujda's School Hospital $SH2$ as two collaborating organizations that share certain resources and services to achieve a common goal. $SH1$ and $SH2$ utilize cloud-based services provided by Azure and Amazon, to outsource their IT infrastructure and software, including computing, networking, and data storage.

Within this particular use case, in Table I, we consider that Azure cloud offers three services, namely $s1$, $s2$, and $s3$, while Amazon cloud provides three services, namely $s4$, $s5$, and $s6$. Each customer cloud creates its own tenants using the services offered by Azure and Amazon clouds. For instance, the customer $SH1$ creates tenants $(t1, t2, t3, t4, t5)$ using the services $(s1, s2, s3, s4, s5)$ provided by Azure and Amazon, respectively. Similarly, the customer $SH2$ creates its own tenants $(t6, t7, t8, t9, t10)$ from the available services $(s4, s5, s6, s1, s2)$, respectively. During collaboration, users

TABLE I. Multi-Cloud use Case

Cloud customers	Cloud providers	Tenants	services
$SH1$	Azure	$(t1, t2, t3)$	$(s1, s2, s3)$
$SH1$	Amazon	$(t4, t5)$	$(s4, s5)$
$SH2$	Amazon	$(t6, t7, t8)$	$(s4, s5, s6)$
$SH2$	Azure	$(t9, t10)$	$(s1, s2)$

from one tenant require access to resources owned by other tenants. Therefore, our scenario gives rise to a set of requirements for cross-tenant access in a multi-cloud environment, which can be categorized into four cases or situations:

- **Case 1:** Collaborating tenants hosted in the same cloud provider and owned by the same cloud customer. For instance, in the Azure cloud, a user from tenant $t1$ requires access to resources owned by tenant $t2$.
- **Case 2:** Collaborating tenants hosted in the same cloud provider and owned by different cloud customers. For example, in the Azure cloud, a user from tenant $t3$ (owned by $SH1$) requires access to resources owned by tenant $t9$ (owned by $SH2$).
- **Case 3:** Collaborating tenants hosted in different cloud providers and owned by the same cloud customer. For example, a user from tenant $t2$ (hosted in Azure cloud) needs access to resources owned by tenant $t5$ (hosted in Amazon cloud).
- **Case 4:** Collaborating tenants hosted in different cloud providers and owned by different cloud customers. For example, a user from tenant $t1$ (owned by $SH1$ and hosted in Azure cloud) needs access to resources owned by tenant $t8$ (owned by $SH2$ and hosted in Amazon cloud).

Our main objective in this approach is to enable secure multi-tenant collaborations in a multi-cloud environment. For this purpose, there is a need for a comprehensive fine-grained access control model that caters to the specific requirements of the scenario.

B. Attribute Based Access Control Model

The following section introduces the ABAC model, which has been tailored to suit the development of MC-ABAC and is not intended to be a comprehensive ABAC model. ABAC has been defined in several ways in the literature, typically for specific use cases. ABAC is an adaptive and a flexible fine-grained access control model.

Definition 1: The core components of ABAC model [8], [22] are:

- U and O represent finite sets of existing users and objects, respectively.
- $A = \{create, read, update, delete\}$ is a finite set of actions.
- $UATT$ and $OATT$ represent finite sets of user and object attribute functions, respectively.
- For each $att \in \{UATT \cup OATT\}$, $range(att)$ represents the attribute's range, which is a finite set of atomic values.
- $attType : UATT \cup OATT \rightarrow \{set, atomic\}$, specifies attributes as set or atomic values.
- Each attribute function maps elements in U to an atomic value or a set
  - $\forall ua \subseteq UATT.\ ua : U \rightarrow Range(ua)\ if\ attType(ua) = atomic$
  - $\forall ua \subseteq UATT.\ ua : U \rightarrow 2^{Range(ua)}\ if\ attType(ua) = set$
- Each attribute function maps elements in O to an atomic value or a set
  - $\forall oa \subseteq OATT.oa : O \rightarrow Range(oa)\ if\ attType(oa) = atomic$
  - $\forall oa \subseteq OATT.oa : O \rightarrow 2^{Range(oa)}\ if\ attType(oa) = set$
- An authorization that decides on whether a user u can access an object o in a particular environment e for the action a, is a boolean function of u, o, and e attributes: Rule: $authorization_a(u, o) \rightarrow f(ATTR(u), ATTR(o))$.

III. Related Work

Several works have been in the literature to ensure access control in multiple environments. In the Task based access control [5] (TBAC), the permissions are granted according to the progress of several tasks. The TRBAC [7] model is constructed by adding the "Task" concept to the RBAC model. In TRBAC, the user has a relationship with permission through role and task. On the other hand, in the Team Access Control Model (TMAC) [6], the permissions are granted to each user through its role and the current activities of the team. These models enable fine-grained access control but they do not incorporate contextual parameters into security considerations and do not support collaboration in multi-domain environment [9], [10]. Moreover, the notion of "Team" used in TMAC model is static. Therefore, this model does not support dynamic collaboration.

Several access control approaches have been proposed to secure resources in cloud environments [11], [12], [13], [15], [16], [23]. Calero et al. [11] introduces a multi-tenancy authorization system based on hierarchical role-based access control with a coarse-grained trust relation and path-based object hierarchies. The work assumes that each issuer may utilize multiple cloud services and collaborate with other issuers.

Another model, Multi-Tenant Role-Based Access Control (MT-RBAC), is proposed by Tang et al. in [13]. This model provides fine-grained access control in collaborative cloud environments by incorporating trust relations among tenants. However, this model does not consider trust relations among issuers separately from tenants. In the MT-RBAC model, the truster exposes certain trusters roles to the trustee, who then assigns their users to these roles. This enables users to access the trusters' resources by activating the trusters' roles. The CTRBAC model [16] extends the traditional RBAC model by introducing new entities to support cross-tenant access and the concept of tasks. However, it should be noted that in this model, a tenant may utilize roles owned by other tenants, which can potentially compromise confidentiality requirements. Additionally, these models are based on the RBAC model, which may lack the necessary flexibility to define complex policy rules.

Several recent approaches have been proposed in [20], [26], [27], focusing on the concept of activity control. This concept expands the scope of traditional access control models by addressing how multiple administrative authorities can effectively collaborate to create, share, manage, and protect digital content and resources. However, these solutions are not suitable for facilitating cross-tenant access in a multi-cloud environment.

The Attribute-Based Access Control (ABAC) model has gained significant attention due to its relevance in addressing the limitations of classical access control models such as RBAC. ABAC, offers a solution that is adaptive and flexible, providing an effective means to describe intricate access control semantics, particularly in collaborative environments. The ABAC model has been extended in several works [14], [17], [28], [29], [30] to support collaboration and resource sharing.

One notable example is the Multi-Tenant Attribute-Based Access Control (MT-ABAC) model [14], which introduces a framework for facilitating collaboration between tenants in a single cloud environment. The MT-ABAC model takes a decentralized approach, allowing each tenant to manage their own access control policies and attributes. However, the MT-ABAC model is primarily designed for multi-tenant environments within a single cloud. These extended models of ABAC may not adequately address access control challenges that arise in multi-cloud environments where users and shared resources span across different tenants in multiple clouds. In such scenario, the complexity increases as different clouds may have their own access control mechanisms and policies.

Authors propose in [19] the authorization federation approach in order to ensure collaboration among organizations whose resources are distributed across multiple cloud service providers. This study primarily focuses on facilitating collaboration among multiple homogeneous clouds, limiting its applicability to heterogeneous multi-cloud environments. Additionally, the emphasis of this approach is on collaboration and resource sharing at the Infrastructure as a Service (IaaS) level, by using openstack cloud [21], overlooking resource sharing at the Platform as a Service (PaaS) and Software as a Service (SaaS) levels.

Therefore, in this paper, we propose the MC-ABAC model, which aims to ensure secure multi-tenant collaborations in a multi-cloud environment. This approach introduces the concepts of tenant, cloud customer, and cloud service provider as key entities in the model. To the best of our knowledge, our work is the first approach that aims to extend the ABAC model to support collaboration in a multi-cloud environment.

IV. Multi-Cloud Attribute Based Access Control Model

In this section, we introduce the MC-ABAC (Multi-Cloud Attributes Based Access Control) model, which extends the ABAC model to support cross-cloud collaboration among multiple clouds. The MC-ABAC model incorporates new entities to facilitate access control of shared resources across multiple clouds. These entities, namely tenant, cloud service provider, and cloud customer, are added to the original ABAC model. Fig. 1 illustrates the structure of the MC-ABAC model. The following sections provide a detailed description of these new entities:

Tenant: is a virtual partition of a cloud service provided by the cloud provider to the customer. The cloud service provider segregates the data and services into multiple tenants. A cloud service provider is defined in this approach as the 5-uplet, (t, U, O, ATT):

- t: The tenant ID;
- U: Set of users belonging to this tenant t;
- O: Set of objects held by this tenant t. An object in cloud could be resource, machine, or service;
- ATT: Set of user, object and environment attributes defined by this tenant t. An object in cloud could be resource, machine, or service;

A cloud service provider (cp) is an organization that offers computing resources, such as storage, platform, application,

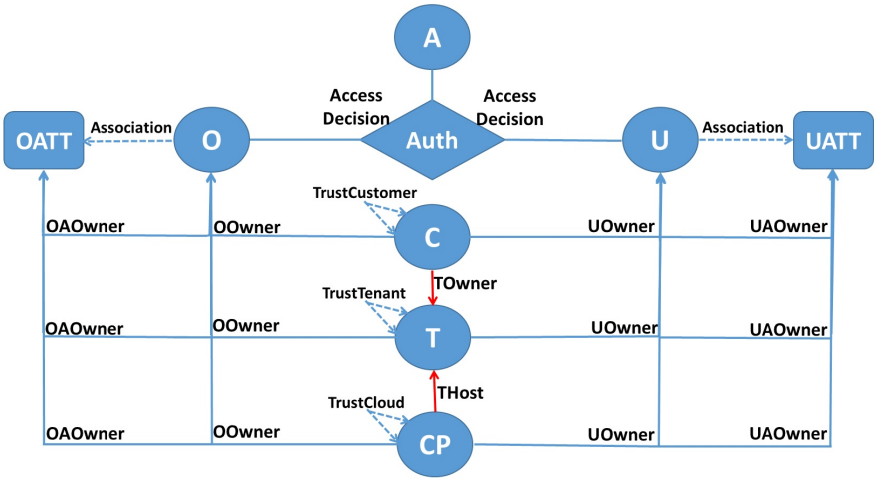

Fig. 1. MC-ABAC model.

and cloud-based compute services, that businesses can access over a network as and when required. These resources can be scaled up or down depending on the customer's needs. A cloud service provider is defined in this approach as the 5-uplet, (cp, U, O, ATT, T):

- cp: The cloud service provider ID;
- U: Set of users belonging to this cp;
- O: Set of objects held by this cp. An object in cloud could be resource, machine, or service;
- ATT: Set of user, object and environment attributes defined by this cp;
- T: Set of tenants hosted in this cp.

A cloud customer (c) refers to an individual or organization that utilizes cloud services. It's worth noting that a cloud customer can be a cloud itself, and that clouds may provide services to each other. A cloud customer is defined as the 5-uplet, (c, U, O, ATT, T):

- c: The cloud customer ID;
- U: Set of users belonging to this c;
- O: Set of objects held by this c. An object in the cloud could be resource, machine, or service;
- ATT: Set of user, object and environment attributes defined by this c.
- T: Set of tenants that this cloud customer c owns.

In the model, every user and object is associated with a unique entiy: Cloud provider, cloud customer or tenant. To achieve this, the model introduces a system-defined attribute called $UOwner$ for users and $OOwner$ for objects. Additionally, each user attribute and object attribute is exclusively owned by a single entity. This is represented using functions $UAOwner$ for user attributes and $OAOwner$ for object attributes.

A. MC-ABAC Definition

Definition 2. Core Multi-Cloud ABAC (Fig. 1) is defined by the basic component sets, functions and authorization policy language given below:

- U, O, T, CP and C represent finite sets of existing users, objects, tenants, cloud service providers and cloud customers respectively.
- A represents a finite set of actions available on objects. Typically $A = \{create; read; update; delete\}$.
- UA and OA represent a finite sets of user and object attribute functions (ua and oa) respectively.
- For each $att \in UA \cup OA$, $range(att)$ represents the attribute's range, which is a finite set of atomic values. An attribute att could be ua or oa.
- $attType : UA \cup OA \rightarrow \{set; atomic\}$, specifies attributes as set or atomic values.
- Each attribute function maps elements in U to an atomic value or a set
  - $\forall ua \in UA.\ ua : U \rightarrow Range(ua)$ if $attType(ua) = atomic$
  - $\forall ua \in UA.\ ua : U \rightarrow 2^{Range(ua)}$ if $attType(ua) = set$
- Each attribute function maps elements in O to an atomic value or a set
  - $\forall oa \in OA.\ oa : O \rightarrow Range(oa)$ if $attType(oa) = atomic$
  - $\forall oa \in OA.\ oa : O \rightarrow 2^{Range(oa)}$ if $attType(oa) = set$
- $UA = GUA \cap LUA \cap CUA$, such as:
  - GUA represents a finite set of global user attribute functions which are defined by the cloud customer C
  - LUA represents a finite set of local user attribute functions which are defined by the tenants T

- CUA represents a finite set of cloud user attribute functions which are defined by the cloud service provider cp

- $OA = GOA \cap LOA \cap COA$, such as:
  - GOA represents a finite set of global object attribute functions which are defined by the cloud customer C
  - LOA represents a finite set of local object attribute functions which are defined by the tenants T
  - COA represents a finite set of cloud object attribute functions which are defined by the cloud service provider cp

- $UOwner : (u : U) \rightarrow E$, required attribute function mapping user u to its owner entity. Note that an entity in this approach, could be a tenant t, a cloud service provider cp or cloud customer c;

- $OOwner : (o : O) \rightarrow E$, required attribute function mapping object o to its owner entity (t, cp or c).

- $UAOwner : (ua : UA) \rightarrow E$, attribute function, mapping user attribute ua to its owner entity (t, cp or c):
  - $ua \in LUA \rightarrow UAOwner(ua) \in T$
  - $ua \in CUA \rightarrow UAOwner(ua) \in CP$
  - $ua \in GUA \rightarrow UAOwner(ua) \in C$

- $OAOwner : (oa : OA) \rightarrow TE$, attribute function, mapping object attribute oa to its owner entity (t, cp or c):
  - $oa \in LOA \rightarrow OAOwner(oa) \in T$
  - $oa \in COA \rightarrow OAOwner(oa) \in CP$
  - $oa \in GOA \rightarrow OAOwner(oa) \in C$

- $TOwner : (t : T) \rightarrow C$, required attribute function mapping tenant t to its owner cloud customer $c \in C$;

- $THost : (t : T) \rightarrow CP$, required attribute function mapping tenant t to its hosting cloud provider $c \in C$;

- The authorizations that decide on whether a user u can access an object o for the action a, are a three Boolean functions of u and o attributes:
  - $LAuth_a(u; o) \rightarrow f(LUA(u); LOA(o))$, represents local authorization that is defined by the tenant;
  - $GAuth_a(u; o) \rightarrow f(GUA(u); GOA(o))$, represents global authorization that is defined by cloud customer;
  - $CAuth_a(u; o) \rightarrow f(CUA(u); COA(o))$, represents cloud authorization which is defined by cloud service provider;

B. Administrative MC-ABAC Model

In this subsection, we focus on the administrative model for the suggested MC-ABAC. This model enables administrators to perform various administrative operations. In this approach, we distinguish three types of administrators: cloud administrator, customer administrator and tenant administrator.

- Cloud administrator: The person who is responsible for defining access policies in the cloud service provider;

- customer administrator: The person responsible for defining access policies in the cloud customer;

- Tenant administrator: The person who is responsible for defining access policies in the tenant entity;

In this model, administrators are granted the ability to perform various administrative operations, with each operation having certain preconditions that need to be satisfied. In the following, we present the formal specification of several administrative operations and their corresponding preconditions:

- $\forall ua \in CUA, ua(u : U)$ is defined by the cloud administrator of cp only if $(UAOwner(ua) = UOwner(u) = cp)$, means that the cloud cp must be the owner of both the user u and the attribute ua. The same principle applies to the subsequent operations as well;

- $\forall oa \in COA, oa(o : O)$ is defined by the cloud administrator only if $(OAOwner(oa) = OOwner(o) = cp)$;

- $\forall ua \in GUA, ua(u : U)$ is defined by the customer administrator of $c \in C$ only if $(UAOwner(ua) = UOwner(u) = c)$;

- $\forall oa \in GOA, oa(o : O)$ is defined by the customer administrator of $c \in C$ only if $(OAOwner(oa) = OOwner(o) = c)$;

- $\forall ua \in LUA, ua(u : U)$ is defined by the tenant administrator of $t \in T$ only if $(UAOwner(ua) = UOwner(u) = t) \cup (TOwner(UAOwner(ua)) = UOwner(u))$, This implies that the user u must be owned by either the tenant t, who is the owner of the attribute ua, or the cloud customer who owns the tenant t.

- $\forall oa \in LOA, oa(o : O)$ is defined by the tenant administrator of $t \in T$ only if $(OAOwner(oa) = OOwner(o) = t)$;

- $CAuth_a(u; o)$ is defined by the cloud administrator of cp. For all attributes ua, oa that are defined in $CAuth_a(u; o)$ only if $UAOwner(ua) = OAOwner(oa) = UOwner(u) = OOwner(o) = cp$. Each authorization function needs to verify that the owner of ua is the same as the owner of oa, u, and o.

- $GAuth_a(u; o)$ is defined by the customer administrator of $c \in C$. For all attributes ua, oa that are defined in $GAuth_a(u; o)$ only if $UAOwner(ua) = OAOwner(oa) = UOwner(u) = OOwner(o) = c$;

- $LAuth_a(u; o)$ is defined by the tenant administrator of $t \in T$. For all attributes ua, oa that are defined in $LAuth_a(u; o)$ only if $(UAOwner(ua) = OAOwner(oa) = UOwner(u) = OOwner(o) = t) \cup (UAOwner(ua) = OAOwner(oa) = OOwner(o) = t \cap UOwner(u) = TOwner(t))$. This predicate $UOwner(u) = TOwner(t)$ means that the

user u is owned by the cloud customer who is the owner of t;

V. Trust Relations

MC-ABAC introduces many trust relations in order to support access between the cloud customer and the cloud service provider, cross tenant access, cross customer access and resource sharing in a multi-cloud environment. These trust relations are established between both, cloud service providers, cloud customers and tenants. In the following, we will define each trust relation used in this model.

A. Provider to Customer Trust Relation

The Provider to customer trust relation $(CPC) \subseteq CP \times C$ is a many-to-many relationship between the cloud service provider cp and the cloud customer c. It is defined as $TrustCPtoC(cp, c) = \cup_i(S_i)$, such as S_i is a cloud service owned by the provider cp. This relationship means that the cloud cp provides a set of services $\cup_i(s_i)$ to the cloud consumer c. This customer c will create its own tenants from these services.

For example, the trust relation $TrustCPtoC(Azure, SH1) = \{s1, s2, s3\}$, means that the customer $SH1$ could create new tenants $t1$, $t2$ and $t3$ from these services $s1$, $s2$ and $s3$, in the cloud cp. Using this trust relation, the cloud consumer can establish or create their own tenants based on the services defined within this relation.

B. Cloud Trust Relation

The cloud trust relation $(CPCP) \subseteq CP \times CP$ is a many-to-many reflexive relation between a truster cloud $cpr \in CP$ and a trustee cloud $cpe \in CP$. It is defined as $TrustCloud(cpr, cpe : CP) \rightarrow 2^T$ means that the cloud service provider cpr authorizes cpr's tenants to collaborate with cpe's tenants. This implies that tenants hosted in cpr could establish tenant trust relation with cpe's tenants, in order to ensure cross tenant access in multi-cloud environment.

This trust relation $TrustCloud(cpr, cpe) = \bigcup_{i=1}^{k} t_i$ is defined with the additional required condition that: $t \in TrustCloud(cpr, cpe)$ only if $THost(t) = cpr$, this implies that the cloud service provider cpr should possess ownership of the tenants which are defined in this relation. For example: *TrustCloud(Azure, Amazon)= t1, t2*, means that tenants hosted in $Azure$ cloud, such as tenants $t1$ or $t2$, can establish a trust relation with tenants from the $Amazon$ cloud.

C. customer Trust Relation

The customer trust relation $(CC) \subseteq C \times C$ is a many-to-many reflexive relation between a truster customer $cr \in C$ and a trustee customer $ce \in C$. It is defined as $Trustcustomer(cr, ce : C) \rightarrow 2^T$, which means that the cloud customer cr authorizes some cr's tenants to collaborate with ce's tenants. This implies that tenants owned by cr could establish tenant trust relation with ce's tenants, in order to ensure cross-tenant access, where each tenant belongs to a different customer, .

This trust relation $Trustcustomer(cr, ce) = \bigcup_{i=1}^{k} t_i$ is defined with the additional required condition that: $t \in Trustcustomer(cr, ce)$ only if $TOwner(t) = cr$, this means that the customer cr must be the owner of tenants which are defined in this relation. For example: *Trustcustomer(SH1, SH2)= t2, t3*, means that tenants belonging to cloud customer $SH1$, such as $t2$ or $t3$, can establish trust relationships with tenants belonging to cloud customer $SH2$.

D. Tenant Trust Relation

The tenant trust relation $(TT) \subseteq T \times T$ is a many-to-many reflexive relation between the truster $tr \in T$ and the trustee $te \in T$. It is defined as $TrustTenant(tr, te : T) \rightarrow 2^U$ which means that the tenant te is authorized to assign values for te's local user attributes to tenant tr's users. This trust relation $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$ is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $UOwner(u) = tr$, this implies that the truster tr must be the owner of users which are defined in this relation. For example: *TrustTenant(t1, t3)= u1, u2*, means that the tenant $t3$ can assign values for $t3$'s local user attributes to $u1$ and $u2$.

This trust relation is subject to a precondition that needs to be satisfied, which depends on the four cases introduced in the case study subsection. In the following, we specify the precondition for each case:

- **Case 1:** Collaborating tenants hosted in the same cloud provider and owned by the same cloud customer. $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$ is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $UOwner(u) = tr$.
- **Case 2:** Collaborating tenants hosted in the same cloud provider and owned by different cloud customers. Before establishing trust between the two tenants tr and te, trust must be established between the two cloud customers cr and ce who are the owners of tr and te respectively. $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$ is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $UOwner(u) = tr \cap tr \in Trustcustomer(TOwner(tr), TOwner(te))$.
- **Case 3:** Collaborating tenants hosted in different cloud providers and owned by the same cloud customer. Before establishing trust between the two tenants tr and te, trust must be established between the two cloud providers cpr and cpe who are the hosts of tr and te respectively. $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$ is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $UOwner(u) = tr \cap tr \in TrustCloud(THost(tr), THost(te))$;
- **Case 4:** Collaborating tenants hosted in different cloud providers and owned by different cloud customers. $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$ is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $UOwner(u) = tr \cap tr \in TrustCloud(THost(tr), THost(te)) \cap tr \in Trustcustomer(TOwner(tr), TOwner(te))$.

We summarize the definition of this trust relation using the following formalism, taking into account the four cases previously discussed: $TrustTenant(tr, te) = \bigcup_{i=1}^{k} u_i$

is defined with the additional required condition that: $u \in TrustTenant(tr, te)$ only if $[(UOwner(u) = tr) \cap [(TOwner(tr) = TOwner(te)) \cup (tr \in Trustcustomer(TOwner(tr), TOwner(te)))] \cap [(THost(tr) = THost(te)) \cup (tr \in TrustCloud(THost(tr), THost(te)))]]$.

Utilizing this trust relation entails redefining the preconditions that correspond to the two operations $oa(o : O)$ and $LAuth_a(u; o)$. Below, we provide the formal specification of these two operations:

- $\forall ua \in LUA, ua(u : U)$ is defined by the tenant administrator of $t \in T$ only if $(UAOwner(ua) = UOwner(u) = t) \cup (TOwner(UAOwner(ua)) = UOwner(u)) \cup (u \in TrustTenant(UOwner(u), UAOwner(ua)))$;
- $LAuth_a(u; o)$ is defined by the tenant administrator of $t \in T$. For all attributes ua, oa that are defined in $LAuth_a(u; o)$ only if $(UAOwner(ua) = OAOwner(oa) = UOwner(u) = OOwner(o) = t) \cup (UAOwner(ua) = OAOwner(oa) = OOwner(o) = t \cap UOwner(u) = TOwner(t)) \cup (UAOwner(ua) = OAOwner(oa) = OOwner(o) = t \cap (u \in TrustTenant(UOwner(u), UAOwner(ua))))$.

VI. Implementation

A. System Architecture

As shown in Fig. 2, we present the implementation architecture of the MC-ABAC model using pyhton technology in order to demonstrate its feasibility. This architecture is composed of five components: Entity information, entity attributes, authorizations, trust relations and policy decision component. In the following, we provide a description of each of these components:

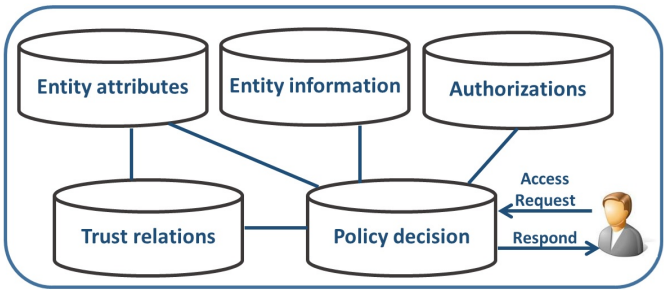

Fig. 2. Implementation architecture.

- **Entity information:** In this component, the cloud administrator, customer administrator, and tenant administrator are responsible for defining and managing various entities. These entities encompass customers, tenants, cloud users and objects, global users and objects, as well as local users and objects. Each administrator is responsible for their respective set of entities within the system;
- **Entity attributes:** Within this component, it is the administrator's responsibility to define the attributes associated with users and objects. These attributes can be categorized as either global, local, or specific to the cloud provider. Subsequently, the administrator assigns values to the user and object attributes for their respective users and objects. This is achieved through a function that takes the user or object as input and generates a value from the range of the attribute. For instance, $attr1(user1) = \{val1, val2\}$: means that for the user $user1$ the values of the attribute $attr1$ are $val1$ and $val2$;
- **Authorizations:** In this component, the cloud administrator, customer administrator, and tenant administrator are responsible for specifying the authorizations policy $CAuth_a(u; o)$, $GAuth_a(u; o)$ and $LAuth_a(u; o)$. In our approach, we consider that each cloud, customer and tenant defines its own policy rules. Note that at this level, we assume that the security policy rules are valid and free from conflicts.
- **Trust relations:** Within this component, the management of trust relations is handled. These trust relations, namely $TrustCloud()$, $TrustCustomer()$, and $TrustTenant()$, are established by the cloud administrator, customer administrator, and tenant administrator, respectively. Moreover, this component governs the trust between the cloud provider and the customer by using the trust relation $TrustCPtoC()$.
- **Policy decision:** In this component, the evaluation of access requests to objects stored in the cloud is carried out based on the collected attribute values and authorizations. When a user submits a request to access a resource within the cloud, the policy decision component assesses the request against the policy rules to determine whether the user has the authorization to access the requested resource or not.

B. Results and Performance

The experiments were conducted on a virtual machine with the following specifications: Memory: 4096 MB, CPU: 2 cores, Hard Disk: 30 GB. For the analysis, we have used a synthetic dataset, containing up to 1000 authorizations, 200 attributes, 2000 attribute assignments and 25 tenants. We observed that the performance of our approach is influenced by various factors, including the number of autorizations and attribute assignments. The evaluation results indicate that the implementation of the MC-ABAC model for defining access control policies incurs minimal overhead.

The average time to grant the access to an object (Fig. 3(a)) with ABAC model increases with 13.1% and 28.5% for policies of 200 and 1000 rules respectively using the MC-ABAC model. The waiting time that is required to get a policy decision increases when there are too many authorizations to be collected. We attest that our implementation performs well, even for a large number of authorizations.

Furthermore, the running time for access/deny decisions to an object has been computed, using both ABAC and MC-ABAC models, for 600 authorizations and for 400 to 2000 attribute assignments. Fig. 3(b) demonstrates that the average time to access a resource with ABAC model increases with

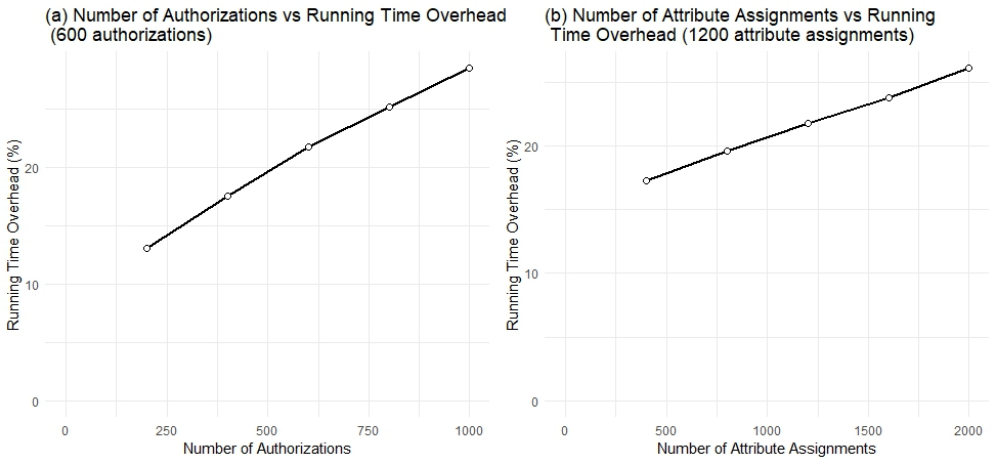

Fig. 3. Running time overhead for access/deny decisions.

17.3% and 26.1% for 400 and 2000 attribute assignments using MC-ABAC Module. We recognize that the implementation of our model demonstrates effective performance when handling a significant number of attribute assignments.

VII. Conclusion

In this paper, we present a novel ABAC model called: MC-ABAC, that leverages the capabilities of ABAC, such as flexibility and adaptability. MC-ABAC enables collaboration and resource sharing among tenants, which are hosted across multiple cloud providers. This model introduces the notions of tenant, cloud customer and cloud service provider as fundamental entities within the model. Moreover, MC-ABAC integrates various trust relations to facilitate effective collaboration and cross-tenant access in the multi-cloud environment. This model is designed to address access control challenges that arise in heterogeneous multi-cloud environments where users and shared resources are distributed across different tenants in multiple clouds.

Finally, the implementation architecture of the MC-ABAC model using Python technology has been proposed to demonstrate its feasibility. The evaluation results have demonstrated that implementing this model for defining access control policies has minimal overhead. As a future work, we plan to implement and test MC-ABAC model on a real platform consisting of multiple cloud providers. Another future research involves developing a solution to detect and resolve conflicting rules in access policies that are defined using the MC-ABAC model.

References

[1] A. Tanvir and A. R. Tripathi, *Specification and verification of security requirements in a programming model for decentralized CSCW systems*, ACM Trans. Inf. Syst. Secur. 10(2) (2007).

[2] P. Mell and T. Grance, *The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (Draft)*, Retrieved September 10, 2011, from http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145-cloud-definition.pdf.

[3] Jin, X., Krishnan and R., Sandhu, *A unified attribute-based access control model covering DAC, MAC and RBAC*, DBSec 12, p. 41-55. 2012.

[4] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman, *Role Based Access Control Models*. IEEE Computer, 29(2), February 1996.

[5] R. Thomas and R. Sandhu, *Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management*, 11th IFIP WorkingConference on Database Security, Lake Tahoe, California, USA, 1997.

[6] R. Thomas, *TMAC: A primitive for Applying RBAC in collaborative environment*, 2nd ACM, Workshop on RBAC, Fairfax, Virginia, USA, P. 13-19, November 1997.

[7] O.H. Sejong and S.Park, *Task-role-based Access Control Model*, In: Information Systems, 28(6): P. 533-562, 2003.

[8] E. Yuan and J. Tong, *Attributed Based Access Control (ABAC) for Web Services*, ICWS IEEE Computer Society, P. 561-569. 2005.

[9] Z. Zhang, X. Zhang, and R. Sandhu, *ROBAC: Scalable role and organization based access control models*, In Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), IEEE, Atlanta, USA, P. 1–9, November 2006.

[10] D. Lin, P. Rao, E. Bertino, N. Li and J. Lobo, *Policy decomposition for collaborative access control*, SACMAT, P. 103-112, 2008.

[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, vol. 8, no. 6, P. 48-55. 2010.

[12] B. Tang and R. Sandhu, *Multi-tenancy authorization models for collaborative cloud services*, in IEEE International Conference on Collaboration Technologies and Systems, P. 132-138, 2013.

[13] B. Tang and R. Sandhu, *A Multi-Tenant RBAC Model for Collaborative Cloud* in PST, P. 229-238, 2013.

[14] N. Pustchi, R. Sandhu, *MT-ABAC: A Multi-Tenant Attribute-Based Access Control Model with Tenant Trust*. NSS. P. 206-220. 2015.

[15] M.A. Madani, M. Erradi and Y. Benkaouz, *Access Control in a Collaborative Session in Multi Tenant Environment* 11th International Conference on Information Assurance and Security, Marrakech, P. 129-134, December 2015.

[16] M. A. Madani, M. Erradi and Y. Benkaouz, *A Collaborative Task Role Based Access Control Model*, Journal of Information Assurance andSecurity, vol. 11, no. 6, P. 348-358, 2016.

[17] M. A. Madani, M. Erradi and Y. Benkaouz, *C-ABAC: An ABAC based Model for Collaboration in Multi-tenant Environment*, Journal of EAI Endorsed Transactions on Smart Cities Volume 2, Issue 8, 26th June 2018.

[18] M. A. Madani, M. Erradi and Y. Benkaouz, *ABAC Based Online Collaborations in the Cloud*, First International EAI Conference, AFRICATEK 2017, LNICST Springer, Marrakech, Morocco, P. 67-76, March 2017.

[19] Navid Pustchi, Ram Krishnan and Ravi Sandhu, *Authorization Federation in IaaS Multi Cloud*, 3rd International Workshop on Security in Cloud Computing, SCC'15, pp. 63–71, April 2015.

[20] M. Gupta and R. Sandhu, *Towards Activity-Centric Access Control for Smart Collaborative Ecosystems*. SACMAT 21, Spain, June 2021.

[21] *OpenStack cloud platform*, http://www.openstack.org/. Accessed: 2023-05-30.

[22] X. Jin, R. Krishnan and R. Sandhu, *Role and attribute based collaborative administration of intra-tenant cloud IaaS*, CollaborateCom. P. 261-274. 2014.

[23] B. Tang, R. Sandhu: Cross-tenant trust models in cloud computing. In: Proc. of Int. Conf. IRI, IEEE, pp. 129–136. 2013.

[24] P. Biswas, R. Sandhu, R. Krishnan, *An Attribute Based Protection Model for JSON Documents*, In NSS, P. 303-317, 2016.

[25] H. Takabi, J. B. D. Joshi, and G. J. Ahn, *SecureCloud: Towards a Comprehensive Security Framework for Cloud Computing Environments*, In Proc. of the 1st IEEE International Workshop Emerging Applications for Cloud Computing, Seoul, South Korea, P. 393-398, 2010.

[26] J. Park, R. Sandhu, M. Gupta and S. Bhat, *Activity Control Design Principles: Next Generation Access Control for Smart and Collaborative Systems*, journal IEEE Access, Volume 9, P. 151004-151022, Novembre 2021.

[27] T. Mawla, M. Gupta and R. Sandhu, *BlueSky: Activity Control: A Vision for "Active" Security Models for Smart Collaborative Systems*. SACMAT 22, New York, USA, pp. 207-216, June 2022.

[28] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei and P. Hon, *An Attribute-Based Controlled Collaborative Access Control Scheme for Public Cloud Storage*. IEEE Transactions on Information Forensics and Security, Volume: 14 Issue: 11, PP. 2927 - 2942, April 2019.

[29] M. Gupta and R. Sandhu, T. Mawla and J. O. Benson, *Reachability Analysis for Attributes in ABAC With Group Hierarchy*. IEEE Transactions on Dependable and Secure Computing VOLume 20, issue 1, PP. 841-858, 2023.

[30] K. Liu, C. Wang and X. Zhou, *Decentralizing access control system for data sharing in smart grid*. High-Confidence Computing, Volume 3, Issue 2, 2023.