ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/283624667

AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing

Conference Paper · October 2015

DOI: 10.1109/CIC.2015.38

CITATION
1

READS
12,197

4 authors, including:

Khaled Riad
University of Science and Technology Beijing
15 PUBLICATIONS **16** CITATIONS

SEE PROFILE

All in-text references underlined in blue are linked to publications on ResearchGate, letting you access and read them immediately.

Available from: Khaled Riad
Retrieved on: 12 September 2016

AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing

Khaled Riad*†, Zhu Yan*, Hongxin Hu‡ and Gail-Joon Ahn§
*School of Computer and Communication Engineering, University of Science and Technology Beijing
P.O. Box 100083, Beijing, China. Email: khaled.riad@zu.edu.eg and zhuyan@ustb.edu.cn
†Mathematics Department, Faculty of Science, Zagazig University
P.O. Box 44519, Egypt. Email: khaled.riad@zu.edu.eg
‡Division of Computer Science School of Computing, Clemson University
SC 29634, Clemson, USA. Email: hongxih@clemson.edu
§School of Computing, Informatics and Decision Systems Engineering, Arizona State University
AZ 85281, Tempe, USA. Email: gahn@asu.edu

***Abstract*—One of the most important challenges that have threatened cloud computing and caused its slow adoption is security. Since clouds have diverse groups of users with different sets of security requirements, restricting the users' accesses and protecting information from unauthorized accesses have become the most difficult tasks. To address these critical challenges, in this paper we first formalize Attribute Based Access Control (ABAC) and propose a new access control model, called Attribute-Rule ABAC (AR-ABAC), for cloud computing to meet critical access control requirements in clouds. Our model supports the attribute-rules that deal with the association between users and objects, as well as the capability for accessing objects based on their sensitivity levels. The attribute-rules specify an agreement that determines what kind of attributes should be used and the number of attributes considered for making access decisions. In addition, our model ensures secure resource sharing among potential untrusted tenants and supports different access permissions to the same user at the same session.**

I. Introduction

Although cloud computing brings many benefits, it may suffer from conventional distributed systems' security attacks [22]. Moreover, a cloud has brought new concerns such as moving resources and storing data in the cloud which may reside in another country that should fulfill different regulations. In this paper, We focus on addressing access control issues in cloud Infrastructure-as-a-Service (IaaS).

Access control is an essential mechanism that controls what operations the user may or may not be able to do. The basic goal of any access control system is to restrict a user to exactly what s/he should be able to do and protect information from unauthorized access. A robust access control model has to cope with some basic issues due to the cloud computing nature, such as:

- Pooling the cloud resources to serve a large number of users with different classifications that can handle diverse permissions associated with the same cloud user;
- Giving the user the ability to use multiple services with respect to authentication and login time;
- Transferring users' credentials across layers to access services and resources; and
- Using multi-tenancy where different resources are dynamically allocated and de-allocated on demand while the location of each resource is being unknown.

All of these facts clearly indicate the necessity of an effective access control model for cloud computing.

Our motivation is to formalize the Attribute Based Access Control (ABAC) that accommodates some special objectives. It should be mentioned that there is no widely accepted formal ABAC model as there are for DAC, MAC and RBAC [4]. We utilize a working ABAC definition stated by NIST special publication 800-162 [9], to state the formal ABAC definition. Also, We propose a new access control model for cloud computing based on the formal ABAC model, called Attribute-Rule ABAC (AR-ABAC). AR-ABAC ensures secure resource sharing among potential untrusted tenants and supports different access permissions to the same user at the same session. Also, our model is flexible enough to support a set of constraints that represent the basic requirements for the cloud access control model. Finally, compared with existing cloud access control models, our model has enough flexibility to cope with different access permissions for the same user. Specifically in this paper:

- We present a formal definition of ABAC that accommodates some special objectives.
- We propose a new access control model for cloud computing (AR-ABAC), our proposed model has a set of features that distinguish it from other traditional and current models proposed for cloud computing:
  i. We propose the Attribute-Rule (AR) to define an agreement on what kind of attributes should be used and how many attributes should be taken into account for making access decisions, as discussed in Subsection IV-A.

978-1-5090-0089-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CIC.2015.38

IEEE computer society

ii. The model assigns each user a specific role based on the possessed attributes. Each role is assigned a set of tasks. Each task inherits its power from its assigned role, and hence each task is assigned a set of permissions so that it can access an object.
iii. Our model can utilize the cloud computing's Infrastructure-as-a-Service (IaaS), and provide four different ways to access it, as described in Subsection V-B.

- We validate our proposed model through experiments with an open-source OpenStack cloud platform as elaborated in Section V.

This paper is organized as follows: Section II provides an overview for the traditional access control models and their abilities to be applied for cloud computing. Section III describes the formal definitions of the ABAC model and ABAC features. Section IV presents our proposed AR-ABAC model. The model analysis, implementation and experimental verification are presented in Section V. The related cloud based access control models are summarized in Section VI. This is followed by our conclusion and future work in Section VII.

II. Traditional Access Control models and their Abilities to be Applied for Cloud Computing

Each of the traditional access control models was proposed for a specific environment with a set of basic requirements:

MAC Model [5]: Mandatory Access Control (MAC) model, where a central authority is in command of giving access decisions to a user/subject requesting access to objects. MAC provides protection against information flow and indirect information leakages, but does not guarantee complete secrecy of the information. Also this model is very expensive and difficult to deploy and does not support: separation of duties, least privilege, and delegation or inheritance principles. Also dynamic activation of access rights for certain tasks is not supported. Moreover, it does not support time and location constraints.

DAC Model [11]: Discretionary Access Control (DAC) model grants the owners of objects the ability to restrict access to their objects, or information in the objects based upon users' identities or a membership in certain groups. DAC model is generally less secure than MAC model, so it is used in environments that do not require a high level of protection [8]. DAC has many side-effects when it is utilized in cloud computing, for example, it does not have the ability to control information flow or deal with Trojan horses that can inherit access permissions [15]; a user may pass her rights to another user, and that can violate the integrity and confidentiality of objects; and finally, it is not scalable enough for cloud computing.

Hierarchical RBAC Model [16]: Role-Based Access Control (RBAC) model is considered as a natural way to control access to resources in organizations and enterprises. The motivation behind RBAC comes from considering "a subject's responsibility is more important than whom the subject is".

RBAC fails to cope with the following issues: the dynamic/random behaviors of users; it also does not consider the time and location constraints; it does not support active responsibilities as it does not separate tasks form roles; it has to deal with a lack of sophisticated semantic models to represent and communicate privileges; and before utilizing the RBAC in cloud computing, it has to ensure granting access decisions in a reasonable time.

ABAC Model [3]: Attribute Based Access Control (ABAC) model relies on a set of attributes associated with a requester or a resource to be accessed in order to make access decisions. There are many ways to define or use attributes in this model. An attribute can be a user's work start date, a location of a user, a role of a user, or all of them. Attributes may or may not be related to each other. However, reaching an agreement about what kind of attributes should be used, and how many attributes are taken into account for making access decisions is a complex task in cloud computing [10]. Finally, proposing a security policy that can work accurately with the ABAC model is vital, because the security policy is responsible for selecting appropriate attributes that are utilized to make correct access decisions.

Risk-BAC Model [6]: Risk-Based Access Control (R-BAC) was proposed by Brucker et al. to cope with multinational organizations that face various kinds of policies and regulations. R-BAC uses different kinds of risk levels with environmental conditions and utilizes the principle of "operational need" to make access decisions. However, R-BAC is difficult to be deployed in cloud computing because of the amount of analysis required and the number of systems to be merged to compute risk levels. It needs expertise that can deal with the model efficiently. Finally, security policies and environmental conditions need to be standardized as they play a crucial role on making access decisions.

III. Formal ABAC Model

There has been considerable prior work for ABAC in various aspects. An early paper on web services states that ABAC "grants accesses to services based on the attributes possessed by the requester" [20], while [7] describes ABAC as an approach, in which "attribute values associated with users determine the association of users with privileges".

Finally a high level definition of ABAC defined by NIST special publication 800-162 [9] that is "An access control method where a subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions". The basic components of ABAC are defined as follows:

- Attributes (A), are characteristics of the user, subject, object, or environment conditions. Attributes contain information given by a name-value pair: $A = \{\{UA\}, \{SuA\}, \{OA\}, \{EcA\}\}$ - is a finite set of user, subject, object, and environment condition attributes respectively.

Where for each attribute (A^*) in $UA \cup SuA \cup OA \cup EcA$, there exists $Range^*_A$, a constant finite set of atomic values. Then each attribute can be either atomic or set of values. Hence the attribute values are mapped dynamically based on the following function $A^* = \{^{Range^*_A \quad if\ attType(A^*)=atomic}_{2^{Range^*_A} \quad if\ attType(A^*)=set}$

The purpose of such detailed categories of attributes is to achieve an effective fine-grained access control. We employed the tables of database to represent and implement the three aforementioned categories of attributes.

- Environment Conditions ($E-Cond$), are operational or situational context in which access requests occur. $E-Cond$ is represented by a finite set of attributes (EcA).
- Users (U) - is a set of human users: $\forall u_i \in U \rightarrow \exists$ a finite set of user attributes ($u_iA \subseteq UA$), that represents u_i's identity. Each user can create multiple subjects and each subject inherits it's attributes from the creator user.
- Subjects (Su) - is a set of non-person entity (the entity requesting to perform permissions upon objects): $\forall su_i \in Su \rightarrow \exists$ a creator user u_{ci} with a set of attributes $u_{ci}A$, that create su_i, where $\forall su_i \rightarrow \exists$ a finite set of subject attributes ($su_iA \subseteq u_{ci}A$), that represents su_i identity.
- Objects (O), is a set of system resources, such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information, for which access is managed.
  $\forall o_j \in O \rightarrow \exists$ a finite set of object attributes ($o_jA \subseteq OA$), objects are assigned their attributes by their creators. The objects can be created by a subject and the initial values of the objects' attributes are assigned by the creator subject.
- Permissions (P), can be known as authorization, access right, or privilege. It is the execution of a function at the request of a subject upon an object.
- Policy (Po), is the representation of rules or relationships that makes it possible to determine if a requested access should be allowed, given the values of the attributes of the user/subject, object, and possibly environment conditions and constrains.

IV. Attribute-Rule ABAC (AR-ABAC)

Our model uses the aforementioned formal ABAC definition (Section III) as the backbone as well as the basic features and concepts of ABAC, as follows:

- ABAC relies upon the evaluation of the subject attributes, the object attributes, environment conditions, and the formal relationship/access control rule or policy defining the allowable operations for subject-object attribute sets.
- Each subject that uses the system must be assigned with specific attributes. These subject attributes are assigned and managed by an authority within the organization that maintains the subject identity information.
- ABAC relies upon the assignment of attributes to subjects and objects, and the development of policy that contains the access rules. Each object within the system must be assigned specific object attributes that characterize the object.
- each object within the system have at least one policy that defines the access rules for the allowable subjects, operations, and environment conditions to the object.
- The rules that bind subject and object attributes indirectly specify privileges (i.e., which subjects can perform which operations on which objects).
- Once subject attributes, object attributes, and policies are established, objects are protected using ABAC. Access control mechanisms mediate access to the objects by limiting access to allowed operations by allowed subjects.

Any access control model going to be deployed in cloud computing has a set of mandatory requirements to be reached. Our proposed model (AR-ABAC) can fulfill a set of cloud computing access control requirements, such as:

- Flexibility in attribute management.
- Least of permissions.
- Supporting users' tenant management.

This can be done using the aforementioned formal ABAC as well as ABAC features, and defining more extra components in our model, where each component supports at least one access control requirement for cloud computing, as follows:

A. Flexibility in Attribute Management

Since reaching an agreement about what kind of attributes should be used, and number of attributes should be taken into account for making access decisions is a complex task [10]. In our model We propose the Attribute-Rule (AR) to overcome this issue. Where AR can be simply defined as a set of predefined rules for both users and objects. AR is basically interested in the user and object attributes. Where AR can weight each attribute and each set of attributes. The weight of each attribute indicates its power to be used, and the weight of each set of attributes reflects its power to be taken into account as well as its assigned role/sensitivity level. In detail AR is defined based on a sequence of steps:

Fig. 1. The sets of attributes' average weight classification groups in our model, where each group is represented by its lower and upper bounds.

1) **Weights of attributes,** the organization assigns each attribute a single weigh value ($w \in [0, Max_w]$). This weight reflects the attribute power and importance. For all attributes, including selected and unselected attributes, there is a set of selected and others unselected attributes that will be used in the model according to the organization. In our model we consider the zero weight attributes are unselected. The unselected attributes are

not used in the next steps because they do not affect the model for their zero weight, such as the Notes attribute.

2) **Attributes' sets power,** the average weight for each set of attributes reflects the set's power. The sets average weights are classified into groups, as shown in Figure 1. The figure illustrates five classification groups for the average weight (A_w) of the sets of attributes. From bottom to top: **G1.** represents the lowest powerful group, for a set of attributes with average weight A_w to be in G1., A_w must satisfy ($L_w \preceq A_w \prec BM_w$) where $L_w \succ 0$ because zero represents the unselected attributes; **G2.** is more powerful than G1., for a set of attributes with average weight A_w to be in G2., A_w must satisfy ($BM_w \preceq A_w \prec M_w$); **G3.** represents the medium powerful group, for a set of attributes with average weight A_w to be in G3., A_w must satisfy ($M_w \preceq A_w \prec AM_w$); **G4.** is more powerful than G3., for a set of attributes with average weight A_w to be in G4., A_w must satisfy ($AM_w \preceq A_w \prec H_w$); and **G5.** represents the highest powerful group, for a set of attributes with average weight A_w to be in G5., A_w must satisfy ($H_w \preceq A_w \preceq max_w$);
3) **Expected role/sensitivity level,** based on the set average weight classification, there should be a specific role/ sensitivity level that must be assigned for each set of attributes.
4) **Union of attributes' sets,** this item is only related to the user attributes, because the user can be assigned multiple roles, but the object is only assigned one sensitivity level. In case of users, the user can be assigned with multiple sets of attributes, hence the user is assigned to the associated roles to these attributes' sets.

Hence the attribute-rules ($AR = \{\{UR\}, \{OR\}\}$) can be summarized as follows:

- User-Rules (UR) - is a set of some sets of user attributes, where each set of user attributes is internally assigned one classification group from Figure 1 and then can be assigned to a user: $UR = \{\{ur_1\}, \cdots, \{ur_n\}\} | ur_i = \{A_1, \cdots A_i\} \subset UA$. UR reflects internally the associated average weight classification group, and hence the associated role for each set of user attributes ur_i.
- Object-Rules (OR) - is a set of some sets of object attributes, where each set of object attributes is internally assigned one classification group from Figure 1 and then can be assigned to an object: $OR = \{\{or_1\}, \cdots, \{or_n\}\} | or_j = \{A_1^{'}, \cdots A_j^{'}\} \subset OA$. OR reflects internally the associated average weight classification group, and hence the associated sensitivity level for each set of object attributes or_i.

A simple example for the sets of user/object attributes is shown in Figure 2, where the figure illustrates that there are some unused, shared, and unshared attributes between different sets. The unused attributes have zero weight (red color points), such as the Notes attribute. The shared attributes between multiple sets (blue color points) represents the common attributes, such as ID or ProjectID. The used but unshared attributes (green color points) represent the most important attributes for that set, and they can control the overall weight of that set, such as HeadManager or DepManager attributes. Each set of attributes inherits its power from the possessed attributes, which are reflected by the average weight of the set and then the assigned classification group from Figure 1.

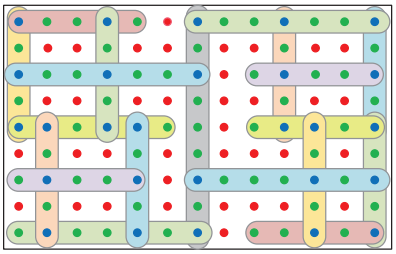

Fig. 2. A simple example for the user/object sets of attributes in our model (user-rules/object-rules).

The attribute-rule can set an agreement about a kind of attributes used and also the number of attributes that can represent each role/sensitivity level, and make access decisions based on the possessed attributes for both users and objects according to the role power for users and the level of sensitivity for objects.

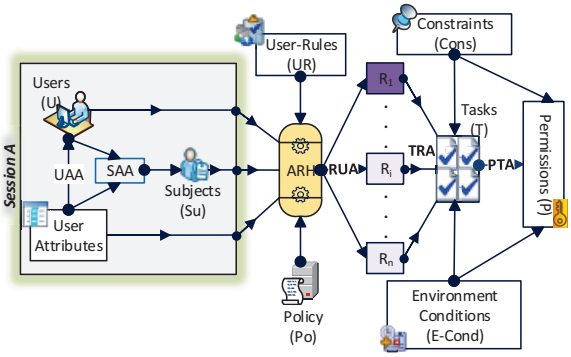

Fig. 3. The attribute-rule attribute based access control ($AR-ABAC$) model users and subjects' side.

B. Assign and Ease of Privileges

The roles, tasks, and permissions assignment processes are based on the Attribute-Role Hierarchy (ARH) - $ARH \subseteq R \times R$ is a partial order on R called the role hierarchy or role power relation, written as $\preceq$. It is based on the user-rules (UR), and also based on some assignment functions:

- Role User Assignment (RUA) - it is a function used to assign each user a set of roles, based on the possessed attributes by the user and the user-rules (UR): $\forall u_i \in U \rightarrow \exists \{r_i\} \subseteq R$ that can be assigned to u_i.
- Task Role Assignment (TRA) - it is a function used to assign each role a set of tasks, based on the user-rules (UR): $\forall r_i \in R \rightarrow \exists \{t_i\} \subseteq T$ that can be assigned to each $r_i \in \{r_i\}$.

- Permission Task Assignment (PTA) - it is a function used to assign each task a set of permissions, based on the user-rules (UR), object-rules (OR): $\forall t_i \in T \rightarrow \exists\{p_i\} \subseteq P$ that can be assigned to each $t_i \in \{t_i\}$.

It should be mentioned that the role user assignment (RUA), task role assignment (TRA), and permission task assignment (PTA) are done internally and dynamically using other helping factors (user-rules (UR), policy (Po), constraints ($Cons$) and environment conditions ($E-Cond$)) without the user or administrator interference, so it is very fast and the possibility of error seems to be zero, as shown in Figure 3. Where the user-rules (UR) is sets of user attributes as defined in Subsection IV-A, and the constraints ($Cons$) represents a set of rules that regulate and manage the relationships between different entities at the same location as stated in Subsection IV-B1. Also the policy (Po) represents the rules or relationships that determine if a requested access should be allowed or not. Finally, the environment conditions ($E-Cond$) represents the operational/ situational context in which access requests occur, as defined in Section III.

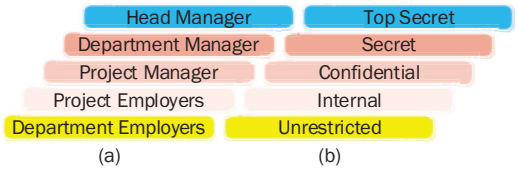

Fig. 4. A simple example for: (a) An organization roles and its power level, and (b) Objects sensitivity levels and its secrecy.

A simple example for an organization roles is shown in Figure 4(a), the figure illustrates five different roles and its power increases from bottom to top.

1) Constraints for Additional Functions: Constraints are an important aspect of role-based access control (RBAC) and are often regarded as one of the principal motivations behind RBAC. The concept of constraints is widely described in [1], [2]. It is possible to distinguish different types of constraints (static or dynamic) that can be attached to the model elements. Our model supports the role concept as the RBAC but the roles are given based on the possessed attributes. In cloud computing constraints can represent a set of basic requirements for any access control model going to be applied. It should be mentioned that constraints depend on the environment conditions such as the current threat level or the authentication time.

For example, the constraints can restrict the user to a set of permissions to do not have an absolute permission. If a user has a permission to stop the virtual machine (VM), so this user can stop any VM, by this way it will be very harmful. The goal of constraints is to restrict the user to stop a specific set of VMs although the user has a general permission to do. Our model supports a set of constraints($Cons$) that can be used at different situations as follows:

- Least of Permissions (LoP): It is the ability of granting subjects the only needed permissions p to accomplish their task t, even when the subject has more permissions than necessarily required for accomplishing tasks: $\forall u_i \in U \rightarrow \{r_1, \cdots, r_n\}, \forall r \in R \rightarrow \{t_1, \cdots, t_n\}$ and $\forall t \in T \rightarrow \{p_1, \cdots, p_n\}$. u_i assigns r_i which uses t_i. And t_i only needs p to accomplish its duty although it has more permissions.
- Delegation of Capabilities (DoC): For flexibility and having dynamic object management in the cloud environment, the delegation of tasks is supported. However, it can only happen between two users/ subjects which have equivalent roles and work within the same area: if $\exists u_i, u_j \in U; r_x \in R$ and u_i is assigned a task $t_x \in r_x$ but can not finish it, then the administrator can delegate t_x to $u_j \leftrightarrow u_i, u_j \in r_x$ and in same location.
- Separation of Duties (SoD): It is basically defined in [2]. It aims at partitioning tasks and permissions associated to roles in order to prevent granting too much authority to one user. It also prevents the conflict of roles and interests. Dynamic SoD is supported in our model for either tasks or roles: if $\exists u_i \in U; r_i, r_j \in R; t_i, t_j \in T | r_i \neq r_j$ and $t_i \neq t_j$, then u_i can activates r_i and $r_j \leftrightarrow r_i \cap r_j = \phi$ and if r_i assigned to u_i; u_i can activates t_i and $t_j \leftrightarrow t_i \cap t_j = \phi$.
- File Syncing and Sharing (FSS): These two services introduce new features for enterprise file sharing solution for online collaboration and storage:
  - File Syncing: It is a new online backup mechanism for syncing data across multiple devices, such as personal computers, tablets or smart phones, as well as collaboration and working with teams.
  - File Sharing: It allows the users to not only access files anywhere, anytime and from a variety of end-point devices, but also collaboratively edit the files together.

Each of the supported constraints represents one of the basic requirements for the cloud computing access control model.

2) Objects and Sensitivity Levels: For the security and privacy of objects, it is necessary for the cloud providers to define various sensitive levels of information in order to restrict who can read and modify information in the cloud. A simple example for a object's sensitivity level and its secrecy is shown in Figure 4(b), where the figure illustrates five sensitivity levels and its secrecy levels increase from bottom to top.

In our model each object ($o \in O$) can be assigned a set of object attributes that represents its sensitivity level: $\forall o_j \in O \rightarrow \exists$ a finite set of object attributes ($o_j A \subseteq OA$), where the sensitivity levels classification and assignment processes are defined as follows:

- Sensitivity Levels (SL) - it is a function used to assign each object a single sensitivity level from a set of security sensitivity levels, based on the object-rules (OR), our model policy and the environment conditions ($E-Cond$), then restrict object's access according to their sensitivity level: $\forall o_i \in O \rightarrow \exists$ a sensitivity level $senl_i \in SenL$ that can be assigned to o_i.

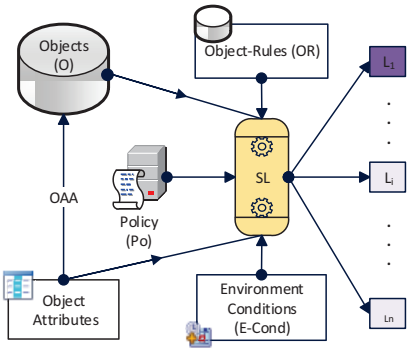

Fig. 5. The attribute-rule attribute based access control ($AR-ABAC$) model objects' side.

- Object Attribute Assignment(OAA) - it is a function used to assign the attribute name and value pairs for each object, based on OR: $\forall o_i \in O \rightarrow \exists$ one or more $or_i \in OR$ that can be assigned to o_i.

Finally, AR-ABAC is also scalable enough to deal with a large number of objects, by classifying them using the sensitivity levels (SL) function, that can classify the objects into different sensitivity levels in a reasonable time, as shown in Figure 5. The figure illustrates the sensitivity levels function that takes as input an object as well as its attributes, and other helping factors like object-rules, policies and environment conditions. And its output is represented in one sensitivity level (from Figure 4) assigned to that object.

V. Implementation and Analysis

A. Implementation on OpenStack

OpenStack is an open-source solution for creating and managing cloud infrastructures [14]. Our next motivation is to implement our model based on our private cloud environment, where our private cloud environment is based on the prominent IaaS platform OpenStack using three physical servers. The configuration of controller node and network node is: 48 cores CPU, 128 GB RAM and 5 TB disk and the configuration of the Nova compute node is: 24 cores CPU, 128 GB RAM and 2 TB disk. Finally, we implemented the AR-ABAC policy-engine on a separate virtual machine which has dual core CPU, 4 GB RAM and 20 GB disk.

B. Experimental Verification

In order to verify our model (AR-ABAC) in cloud IaaS. The cloud IaaS service is represented in the Database server, the Email server and the VM server as well as a set of virtual machines.

Our model can have a considerable impact on controlling access to IaaS. AR-ABAC access policies define roles and tasks needed for each role. Each task is assigned the required permissions to accomplish its job. The AR-ABAC can restrict access to cloud IaaS by four different ways:

- Each object can have its own sensitivity level. Thus, any task attempting to access this object has to have enough power that equals or dominates the sensitivity level of the object to be accessed.
- By restricting access to a number of outlined roles. Therefore, the object do not have sensitivity level and be accessed by the defined roles' members.
- By giving access to the object according to tasks. Therefore, the object does not have sensitivity level and be accessed by the defined tasks only.
- The easiest way is ignoring the object's sensitivity level and allowing access to any authenticated user.

It should be mentioned that the model has no restrictions about ways that can be supported.

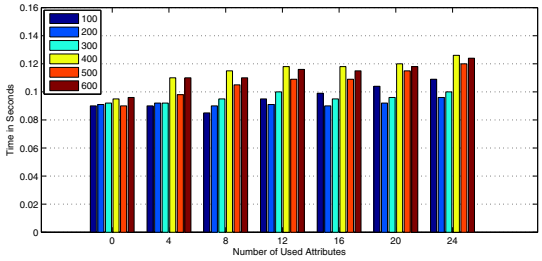

Fig. 6. The average time for token generation in Keystone OpenStack, including and excluding the user attributes.

Finally, we already completed the AR-ABAC integration with the Keystone and Nova services of OpenStack, where the results are represented into two parts:

- **Part one:** Represents the time spent in token (a signed user credential) generation in OpenStack, with and without additional user attributes, as shown in Fig. 6, where including user attributes in the token instead of the role information. This part requires a longer time for token generation. The figure illustrates the response time for token generation using different numbers of attributes (0, 4, 8, 12, 16, 20 and 24), where zero attributes means that the original OpenStack state (just the role attribute), by sending concurrent requests (100, 200, 300, 400, 500 and 600) to keystone and measuring the average response time at the client side. The results show that: **(i)** for the same concurrent request, the average time of token generation increases according to the number of used attributes; **(ii)** the keystone signing and transmission take longer time to finish, but the increase is not significant (about 25% with increasing the number of attributes from 0 to 24.); **(iii)** finally because of the keystone internal scheduling mechanism, at the same number of attributes, the time doesn't increase with the concurrent requests.
- **Part two:** Represents the time spent by Nova service in communicating the AR-ABAC Policy-Engine machine, as shown in Figure 7. The figure illustrates the network latency in communicating the AR-ABAC Policy-Engine using different numbers of attributes (0, 4, 8, 12, 16, 20 and 24), by sending concurrent requests (100, 200, 300,

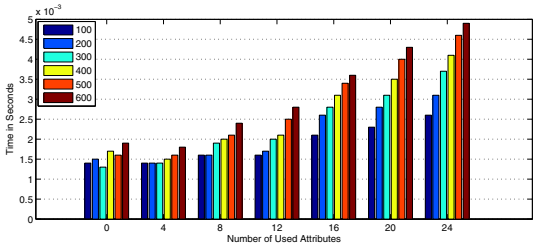

Fig. 7. The average time for Nova communicating the AR-ABAC Policy-Engine machine, including and excluding the user attributes.

400, 500 and 600) from the Nova service to the AR-ABAC Policy-Engine machine. The results show that: **(i)** the latency increases with increasing the number of concurrent requests at the same number of used attributes; **(ii)** the waiting time for getting a policy decision become larger since there are too many requests that have to be evaluated; **(iii)** finally using 24 attributes, the average time for 600 concurrent requests is around 1.8 times of the time of 100 concurrent requests. While for zero attributes, the time for 600 concurrent requests is around 1.3 times of the time for 100 concurrent requests.

In order to validate our proposed model, we have compared it with the conventional and latest proposed access control models for cloud computing. The comparison is based on a set of security features that represents the basic requirements for a cloud computing access control model and either AR-ABAC or other models that can support. It has been found that our model can support the cloud computing access control requirements, such as scalability, heterogeneity, auditing, assign and ease of privileges, flexibility in attribute management, least of permissions, delegation of capabilities, separation of duties, file syncing and sharing, and policy management.

VI. Related Work

Since cloud computing has its own characteristics and features such as mobility and on-demand services, the traditional access control methods can not be used for cloud computing due to several challenges: access control must be dynamic to adapt to the dynamic nature of cloud computing resources joining and leaving; access control should inherit existing security policy among the clouds; in an open cloud computing environment, each resource node may not be familiar (even do not know) each other, identity-based security can not be used; and so on. Hence, cloud providers need a strengthened access control system for controlling admission to their resources with precisely monitoring who accesses them.

There are many access control models presented by researchers for cloud computing. In this section, a brief discussion of various proposed cloud access control models has been presented:

Wang et al. [21]: Suggested an adaptive access algorithm by introducing the trust into cloud computing to decide the access control to the resources using an improved RBAC technique. The trust level is updated and changed automatically by the trust management system according to evolution done by clouds after each transaction.

Tianyi et al. [18]: Proposed coRBAC that is a specifically optimized RBAC system for cloud computing. It inherits the existing RBAC's role model and distributed RBAC's domain model, and optimizes and improves the access control system for services which are hosted on the cloud computing platform. The coRBAC implements an internal RBAC in each organization, and there is only one manager role in each internal RBAC.

Tsai and Shao [19]: Proposed an RBAC model using a role ontology for Multi-Tenancy Architecture (MTA) in clouds. The ontology is used to build up the role hierarchy for a specific domain. In this approach, a subject can have multiple roles in different sessions. Also, a role hierarchy is based on domain ontology and can be transferred between various ontology domains.

Mon and Naing [12]: Proposed a privacy enhancement system on academic-based private cloud system using Eucalyptus open source cloud infrastructure. They attempted to guarantee privacy of cloud's users and security of the personal data, by combining RBAC and ABAC together.

Narayanan and Güne [13]: Adapted Task-Role Based Access Control (T-RBAC) with constraints such as least privilege, separation of duty, delegation of tasks, and spatial and temporal access. Permissions are activated or deactivated according to the current task or process state.

Sun et al. [17]: Presented a semantic based access control model, which considers semantic relations among different entities in cloud computing. They extended the RBAC model using semantic web environments and utilized the semantic scopes of subjects, objects, actions and attributes to define the relations used in ontologies.

Younis et al. [23]: Proposed an access control model for cloud computing called AC3. The AC3 has three different levels of security, which can be used according to the level of trust. It supports various sensitive levels of information in order to restrict who can read and modify information in the cloud.

VII. Conclusion and Future Work

In this paper, we have provided the formal definition of ABAC model, that accommodates some special objectives. We also have proposed a new access control model for cloud computing based on ABAC model, supporting attribute-rule called Attribute-Rule ABAC (AR-ABAC). AR-ABAC can fulfill a set of mandatory requirements for various access control models being deployed in cloud computing, especially cloud IaaS, as concluded here:

- By proposing attribute-rule that can set an agreement about a kind of attributes should be used, and a number of attributes considered for making access decisions, which represents flexibility in attribute management.
- Task power inheritance, where each task inherits its power to access an object based on its assigned roles.

- The model machine can utilize the cloud computing IaaS service, and provide four different IaaS access ways.

The experimental results have shown that AR-ABAC is suitable for cloud IaaS, where the average time for token generation in Keystone including 24 attributes is increased with about 25% more than including zero attributes. Hence the increase is not significant. Also, the average time for Nova communicating the AR-ABAC policy-engine increases by increasing the number of attributes as well as by increasing the number of concurrent requests at the same number of attributes.

Finally, AR-ABAC can fulfill the basic and mandatory access control requirements, which are required for any access control model that is going to be implemented in cloud computing. As our future work, we are going to integrate our proposed model with the Neutron service and other OpenStack services, based on the real private cloud environment.

ACKNOWLEDGMENT

The authors would like to thank the National Natural Foundation of China for partly supporting this work (Grant No. 61472032 and 61170264). Also the work was partially supported by the grants from National Science Foundation (NSF-CNS-1537924, NSF-CNS-1531127 and NSF-IIS-1527421).

REFERENCES

[1] G.-J. Ahn. *The Rcl 2000 Language for Specifying Role-based Authorization Constraints*. PhD thesis, Fairfax, VA, USA, 2000.

[2] G.-J. Ahn and R. S. Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):207–226, November 2000.

[3] M. Al-Kahtani and R. Sandhu. A model for attribute-based user-role assignment. In *18th Annual Computer Security Applications Conference, 2002. Proceedings*, pages 353–362, 2002.

[4] R. Ausanka-Crues. Methods for access control: advances and limitations. *Harvey Mudd College; 2004. Retrieved December 07*, 2012.

[5] D. Bell and L. LaPadula. Secure computer systems: mathematical foundations. *Bedford, MA. Retrieved February 04, 2013, from: Secure computer systems: mathematical foundations; 1973*.

[6] A. D. Brucker, L. Brügger, P. Kearney, and B. Wolffy. An approach to modular and testable security models of real-world health-care applications. In *SACMAT'11. Proceedings of the 16th ACM symposium on Access Control Models and Technologies*, pages 133–142. SACMAT, 2011.

[7] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini. *Collaborative Computing: Networking, Applications and Worksharing*, volume 10 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, chapter A Constraint and Attribute Based Security Framework for Dynamic Role Assignment in Collaborative Environments, pages 322–339. Springer Berlin Heidelberg, 2009.

[8] S. Harris. Mike meyers cissp(r) certification passport. first edition. *United States: McGraw-Hill*, page 422, 2002.

[9] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. Guide to attribute based access control (abac) definition and considerations. Special Publication 800-162, U.S. Department of Commerce, January 2014. National Institute of Standards and Technology.

[10] X. Jin, R. Krishnan, and R. Sandhu. *Data and Applications Security and Privacy XXVI*, volume 7371 of *Lecture Notes in Computer Science*, chapter A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC, pages 41–55. Springer Berlin Heidelberg, 2012.

[11] B. W. Lampson and P. Alto. Acm sigops operating systems review. *SIGOPS ACM Special Interest Group on Operating Systems, ACM New York, NY, USA*, 8(1):18–24, 1974.

[12] E. E. Mon and T. T. Naing. The privacy-aware access control system using attribute-and role-based access control in private cloud. In *4th IEEE International Conference on: Broadband Network and Multimedia Technology (IC-BNMT)*, pages 447–451, October 2011.

[13] H. A. J. Narayanan and M. H. Giine. Ensuring access control in cloud provisioned healthcare systems. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 247–251, January 2011.

[14] OpenStack. http://www.openstack.org/.

[15] P. Samarati and S. C. de Vimercati. *Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, chapter Access Control: Policies, Models, and Mechanisms, pages 137–196. Springer Berlin Heidelberg, 2001.

[16] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for role-based access control: Towards a unified standard. In *5th ACM Workshop on Role-Based Access Control*, pages 47–63. ACM, July 2000.

[17] L. Sun, H. Wang, J. Yong, and G. Wu. Semantic access control for cloud computing based on e-healthcare. In *16th International Conference on: Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE*, pages 512–518, May 2012.

[18] Z. Tianyi, L. Weidong, and S. Jiaxing. An efficient role based access control system for cloud computing. In *11th International Conference on: Computer and Information Technology (CIT), 2011 IEEE*, pages 97–102, Augest 2011.

[19] W.-T. Tsai and Q. Shao. Role-based access-control using reference ontology in clouds. In *10th International Symposium on: Autonomous Decentralized Systems (ISADS)*, pages 121–128, March 2011.

[20] L. Wang, D. Wijesekera, and S. Jajodia. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM workshop on Formal Methods in Security Engineering, FMSE'04*, pages 45–55, ACM, New York, October 2004.

[21] W. Wang, J. Han, M. Song, and X. Wang. The design of a trust and role based access control model in cloud computing. In *6th International Conference on: Pervasive Computing and Applications (ICPCA)*, pages 330–334, October 2011.

[22] Z. Wang. Security and privacy issues within the cloud computing. In *2011 International Conference on: Computational and Information Sciences (ICCIS)*, pages 175–178, October 2011.

[23] Y. A. Younis, K. Kifayat, and M. Merabti. An access control model for cloud computing. *Journal of Information Security and Applications*, 19(1):45 – 60, 2014.