EAR-ABAC: An Extended AR-ABAC Access Control Model for SDN-Integrated Cloud Computing

Khaled Riad
School of Computer and Communication Engineering
University of Science and Technology Beijing
P.O. Box 100083, Beijing, China &
Mathematics Department, Zagazig University
P.O. Box 44519, Zagazig, Egypt

Zhu Yan
School of Computer and Communication Engineering
University of Science and Technology Beijing
P.O. Box 100083, Beijing, China

ABSTRACT

Due to the distinguished nature of cloud computing, it needs an effective access control model, that can cope with its broad network access, on-demand self-service, and so on. When thinking in integrating the Software-defined Networking (SDN) with the cloud computing environment, to let SDN serve, secure, and control the cloud environment. The authors must think about a perfect access control model to secure access to the SDN-integrated cloud environment. This paper proposes an extended access control model for the SDN-integrated cloud computing. Where the author's AR-ABAC access control model [15] is extended to suit the SDN-integrated cloud environment distinguished nature. The extended model can make the election process about the number of attributes considered for making access decisions. In addition it can perfectly deal with the SDN software controllers (OpenDaylight controller). Finally the model ensures secure resource sharing among potential untrusted tenants and supports different access permissions to the same user at the same session.

General Terms

Security, Theory, Access Control Models

Keywords

cloud computing security, software defined networking, attribute based access control

1. INTRODUCTION

Although the cloud computing paradigm fosters significant growth in business productivity with new online services while reducing costs, minimizing risks, and increasing agility; security issues have impacted the cloud wide adoption for enterprises and organizations. Significant innovations in Virtualization and distributed computing, as well as improved access to high-speed Internet and a weak economy, have accelerated interest in cloud computing. Consequently, such trends elevate concerns on cloud security.
The recent rise of the Software-defined Networking (SDN), which is an emerging network architecture where network control is decoupled from forwarding and is directly programmable as defined by the ONF [13]. Multiple network applications have already been proposed and evaluated by the SDN research community. They can be grouped in the areas of network management and traffic engineering, application server load balancing and network access control, SDN security, network Virtualization and inter-domain routing. **In this paper** the authors are interested in access control based on the SDN-integrated cloud computing environment.
Access control is an essential mechanism that controls what operations the user may or may not be able to do. Also access control may monitor and record all attempts made to access a system and identify users with unauthorized attempts to access the system. 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. Cloud computing can be distinguished from other traditional computing models by some major characteristics [9]. Hence the robust access control model has to cope with some basic issues due to the special nature of the SDN-integrarted cloud computing environment, such as:

- On-demand self-service by supporting a pay-as-you-go model;
- Broad network access, where the cloud resources are available to be accessed over multiple device types;
- 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;
- Using multi-tenancy where different resources are dynamically allocated and de-allocated on demand while the location of each resource is being unknown; and
- Programming and managing the forwarding plane using a centralized software controller, while keeping in mind the software differences.

Since a cloud environment involves multiple resources belonging to multiple clients interacting in complex manners, proper access control to these resources is very important. These facts clearly indicate the necessity of an effective access control model for the SDN-integrated cloud computing.
Thinking about the pure Attribute Based Access Control (ABAC) model [1], Where the basic idea is to use characteristics or attributes

of the access requesters as a basis for access decision. Since cloud environments unlike traditional organizations are not static. They are elastic and constantly evolve in response to various factors including the need to support a pay-as-you-go model. This dynamic nature implies that their access control policies need to be adaptable to changing circumstances. Often operational needs may arise that require access be granted under less than ideal security situations. So the pure ABAC cannot adequately model this dynamism.

Motivation: the authors motivation is to extend the AR-ABAC access control model proposed by them earlier [15] in this paper. Hence the extended AR-ABAC can be integrated with the SDN-integrated cloud computing environments. Their extended model ensures secure resource sharing among potential untrusted tenants and supports different access permissions to the same user at the same session. Also, it is flexible enough to support a set of constraints that represent the basic requirements for the SDN-integrated cloud computing access control model. Finally, compared with existing cloud access control models, the extended model has enough flexibility to cope with different access permissions for the same user. Specifically in this paper:

- The AR-ABAC access control model [15] is extended to be integrated with the SDN-integrated cloud computing environment;
- The extended AR-ABAC is a new access control model for the SDN-integrated cloud computing, this model has a set of features that distinguish it from other traditional and current models proposed for cloud computing:
  i. The model can do a dynamic election process to choose the attributes should be used and specify the number of attributes should be taken into account for making access decisions;
  ii. It provides four different ways to access Infrastructure-as-a-Service (IaaS), as described in Subsection 4.2; and
  iii. It can deal with the SDN OpenDaylight [12] controller, and also have the ability to work with other types of controllers such as Floodlight [5].
- The extended AR-ABAC provides the basic operations for the cloud root user and the tenant root user;
- A complete comparison between the extended model, traditional, and proposed access control models, is provided to check the support of the basic requirements for any access control model to be applied for the cloud computing environment; and
- The authors validate proposed model through experiments with an open-source OpenStack cloud platform and the OpenDaylight controller as elaborated in Section 4.

The rest of this paper is organized as follows: Section 2 provides an overview for the traditional access control models and their ability to be applied for the SDN-integrated cloud environment. Section 3 presents an overview about the AR-ABAC access control model proposed by the authors earlier [15]. The extended model implementation and analysis are presented in Section 4. The related cloud based access control models are summarized in Section 5. This is followed by the conclusion and future extensions in Section 6.

2. 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 [2]: 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 [8]: 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 [6]. 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 [16]; 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 [17]: 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 [1]: 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 [7]. 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 [3]: 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.

3. ATTRIBUTE-RULES ABAC (AR-ABAC) ACCESS CONTROL MODEL OVERVIEW

AR-ABAC is the backbone of this paper access control model that is proposed by Riad et. al [15]. AR-ABAC uses the formal ABAC definition and support the use of a new notion proposed by the authors called the Attribute-Rules (AR), which consists of User-Rules

(UR) and Object-Rules (OR). AR can deal with the association between users and objects, as well as the capability for accessing objects based on their sensitivity levels. Also it can specify an agreement that determines what kind of attributes should be used and the number of attributes considered for making access decisions. In addition, the AR-ABAC model ensures secure resource sharing among potential untrusted tenants and supports different access permissions to the same user at the same session. The model consists of two parts: the users and subjects' side and the objects side.

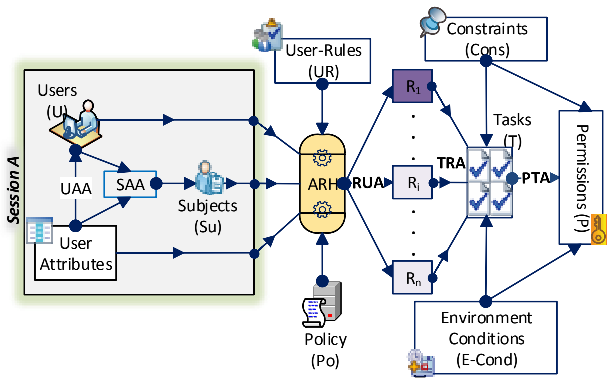

Fig. 1. The attribute-rule attribute based access control ($AR - ABAC$) model users and subjects' side.

In the users and subjects' side (Fig. 1), each user and subject is assigned a set of user attributes using the User Attribute Assignment (UAA) function and the Subject Attribute Assignment (SAA) function respectively. Then there is a set of internal and dynamic assignment functions, for assigning roles, tasks and permissions, as follows:

- 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.

In the objects side (Fig. 2). each object ($o \in O$) can be assigned a set of object attributes that represents its sensitivity level using the Object Attribute Assignment (OAA) function:
$\forall o_j \in O \rightarrow \exists$ a finite set of object attributes ($o_jA \subseteq OA$), where the sensitivity levels classification and assignment processes are defined as follows:

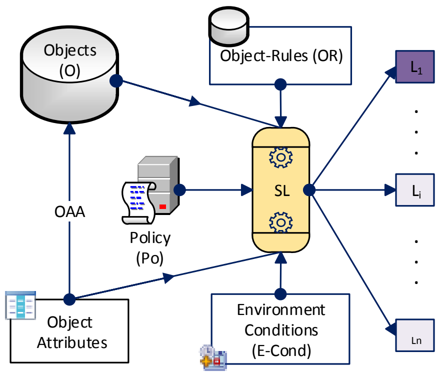

Fig. 2. The attribute-rule attribute based access control ($AR - ABAC$) model objects' side.

- 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), the 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.
- 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.

It should be mentioned that the model object's side is 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.

4. IMPLEMENTATION AND ANALYSIS

4.1 Implementation on OpenStack and OpenDaylight

OpenStack is an open-source solution for creating and managing cloud infrastructures [14]. OpenStack controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with popular enterprise and open source technologies making it ideal for heterogeneous infrastructure. While OpenDaylight (ODL) [12] is a highly available, modular, extensible, scalable and multi-protocol controller infrastructure built for SDN deployments on modern heterogeneous multi-vendor networks. SDN-ODL provides a model-driven service abstraction platform that allows users to write apps that easily work across a wide variety of hardware and south-bound protocols. Providing more details about OpenStack and SDN-ODL is out of the scope of this paper.
The integration of OpenStack and SDN-OpenDaylight (SDN-ODL) is a hot topic. SDN-ODL has driver for Neutron ML2 (Modular Layer 2) plugin to enable communication between OpenStack-Neutron and SDN-ODL. On the SDN controller side, SDN-ODL has northbound APIs to interact with Neutron and use OVSDB (Open vSwitch Database Management Protocol) for southbound configuration of vSwitches on compute nodes. Thus SDN-ODL can manage network connectivity and initiate GRE or VXLAN tunnels for compute nodes.
The author's next motivation is to implement their extended access control model based on their SDN-integrated private cloud environment. Where their private cloud environment is based on the

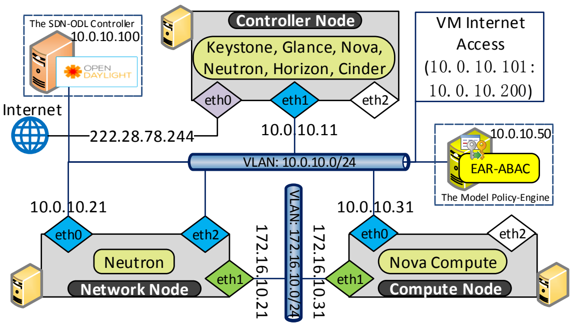

Fig. 3. The author's private cloud environment: OpenStack is installed on three physical machines, the SDN-OpenDaylight controller virtual machine and the EAR-ABAC policy engine virtual machine.

prominent IaaS platform OpenStack and the SDN-OpenDaylight controller, as shown in Fig. 3.
In this figure the OpenStack is installed on three physical machines. The authors install one controller node, one network node and one compute node. 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. There are four networks in this installation: **(i)** VLAN: 10.0.10.0/24 is the management network which connects different components of OpenStack; **(ii)** VLAN: 172.16.10.0/24 is the instance tunnels network which connects the Network and Compute1 nodes; **(iii)** The VM Internet Access (10.0.10.101: 10.0.10.200) network which connects virtual machines with the Internet; and **(iv)** Controller node's eth0 network interface shows the access to the Internet which is only accessible by Controller node, where the Network and Compute node can connect the Internet through the Controller node's eth1 network interface. According to the SDN-ODL controller, the SDN-ODL controller has been implemented on a separate virtual machine connected to VLAN: 10.0.10.0/24, which has dual core CPU, 8 GB RAM, 80 GB disk and eth0 IP:10.0.10.100/24. Finally the EAR-ABAC policy-engine has been implemented on a separate virtual machine which has dual core CPU, 4 GB RAM, 20 GB disk and eth0 IP:10.0.10.50/24.

4.2 Experimental Verification

In order to verify the EAR-ABAC access control model in the SDN-integrated cloud IaaS, as shown in Fig. 4. The figure illustrates: three types of users (cloud root user, tenant root user and normal tenant users). The connection between cloud root user and the tenant root user indicates that the extended model supports multiple tenants and all are under full management by the cloud root user. The connection between the tenant root user and the normal users, indicates that each tenant has full management to it's users. 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. Between the users and the IaaS there is a very important machine, that represents EAR-ABAC policy-engine.
The formal description of the basic functions and operations for both cloud root user and tenant root user is summarized in Table 1. This table illustrates a set of basic operations (24 functions) and its updates, where the cloud root user has three functions and the tenant root user has six categories represented by 21 functions. The API of EAR-ABAC model, that consists of these 24 functions, re-

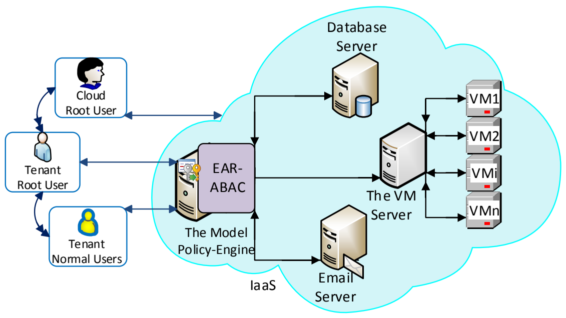

Fig. 4. The AR-ABAC in cloud IaaS, where EAR-ABAC policy-engine is implemented on a separate machine.

alizes a bridge between the tenant root users and cloud IaaS. Since these functions are easy to understand, the tenant root users could interact with the model's API interface in the simplest possible way. The author's extended model can have a considerable impact on controlling access to IaaS. EAR-ABAC access policies define roles and tasks needed for each role. Each task is assigned the required permissions to accomplish its job. The EAR-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 IaaS access ways that can be supported.
Finally, the authors already completed the EAR-ABAC integration with the Keystone, Nova and Neutron services of OpenStack. Also the integration with SDN-ODL is already done. It should be mentioned that when activating the SDN-ODL controller to control the cloud computing environment (SDN-integrated cloud), the communication will be between the model policy engine and the SDN-ODL controller not the OpenStack-Neutron. By using different numbers of attributes (0, 4, 8, 12, 16, 20 and 24). Where zero attributes means that the original OpenStack state. By sending concurrent requests (100, 200, 300, 400, 500 and 600), the results can be represented into four different 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. 5. Where including user attributes in the token instead of the role information, requires a longer time for token generation. The figure illustrates the response time for token generation using different numbers of attributes and sending different concurrent requests 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 takes longer time to finish, but the increase is not significant (about

Table 1. The basic operations for cloud root user and the tenant root user in the extended access control model.

User	Operations	Updates
Cloud Root User	$createTenant(req : CRU, tenant : NAME)$	$T \cup= \{tenant\}$
	$deleteTenant(req : CRU, tenant : T)$	$T \setminus= \{tenant\}$
	$createRootUser(req : CRU, u : NAME, tenant : T)$	$TRU = \{u\}$
Tenant Root User	$createUserAttr(req : TRU, ua : NAME, type : \{atomic, set\})$	$UA \cup= \{ua\} \| attType(ua) = type$
	$createSubAttr(req : TRU, sua : NAME, type : \{atomic, set\})$	$SuA \cup= \{sua\} \| attType(sua) = type$
	$addSubConstr(req : TRU, policy : POLICY)$	$SuConstr \cup= \{policy\}$
	$deleteSubConstr(req : TRU, policy : POLICY)$	$SuConstr \setminus= \{policy\}$
	$UserAttrMod(req : TRU, ua : NAME, value : \{atomic, set\})$	$ua \leftarrow value$
	$SubAttrMod(req : TRU, sua : NAME, value : \{atomic, set\})$	$sua \leftarrow value$
	$UserAttrSync(req : TRU, ua : UA)$	$UA \circlearrowleft= UA$
	$SubAttrSync(req : TRU, sua : SuA)$	$SuA \circlearrowleft= SuA$
	$createObjAttr(req : TRU, oa : NAME, type : \{atomic, set\})$	$OA \cup= \{oa\} \| attType(oa) = type$
	$addObjConstr(req : TRU, policy : POLICY)$	$ObjConstr \cup= \{policy\}$
	$deleteObjConstr(req : TRU, policy : POLICY)$	$ObjConstr \setminus= \{policy\}$
	$addAuthorization(req : TRU, policy : POLICY)$	$ObjConstr \cup= \{policy\}$
	$deleteAuthorization(req : TRU, policy : POLICY)$	$ObjConstr \setminus= \{policy\}$
	$createAdminRole(req : TRU, role : NAME)$	$AdminR \cup= \{role\}$
	$deleteAdminRole(req : TRU, role : NAME)$	$AdminR \setminus= \{role\}$
	$createAdminPolicy(req : TRU, policy : POLICY)$	$AdminP \cup= \{policy\}$
	$deleteAdminPolicy(req : TRU, policy : POLICY)$	$AdminP \setminus= \{policy\}$
	$addAdminUser(req : TRU, u : NAME)$	$TAU \cup= \{u\}$
	$deleteAdminUser(req : TRU, u : NAME)$	$TAU \setminus= \{u\}$
	$addAdminUserRole(req : TRU, u : TAU, r : AdminR)$	$RUA \cup= \{(u, r)\}$
	$deleteAdminUserRole(req : TRU, u : TAU, r : AdminR)$	$RUA \setminus= \{(u, r)\}$

CRU - Cloud Root User, TRU - Tenant Root User, TAU - Tenant Admin Users and RUA - Role User Assignment.

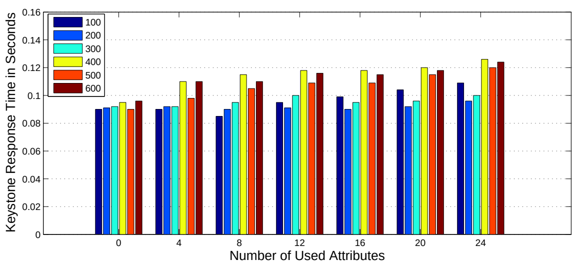

Fig. 5. The average time for token generation in Keystone OpenStack, including and excluding the user attributes.

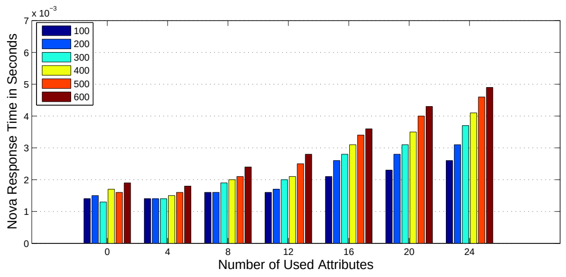

Fig. 6. The average time for Nova communicating the EAR-ABAC Policy-Engine machine, including and excluding the user attributes.

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 do not increase with the concurrent requests.

- **Part two:** Represents the time spent by Nova service in communicating the EAR-ABAC Policy-Engine machine, as shown in Fig. 6. The figure illustrates the network latency in communicating the EAR-ABAC Policy-Engine using different numbers of attributes and sending different concurrent requests from the Nova service to the EAR-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, since there are too many requests have to be evaluated; **(ii)** using 24 attributes, the average time for 600 concurrent requests is around 1.9 times of the time of 100 concurrent requests. While for zero attributes, the time for 600 concurrent requests is around 1.38 times of the time for 100 concurrent requests.
- **Part three:** Represents the time spent by Neutron service in communicating the EAR-ABAC Policy-Engine machine, as shown in Fig. 7 The figure illustrates the network latency in communicating the EAR-ABAC Policy-Engine using different numbers of attributes and sending different concurrent requests from the Neutron service to the EAR-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, since there are too many requests have to be evaluated; **(iii)** using 24 attributes, the average time for 600 concurrent requests is around 2.16 times of the time of 100 concurrent requests. While for zero attributes, the time for 600 concurrent requests is around 1.355 times of the time for 100 concurrent requests.
- **Part four:** Represents the time spent by SDN-ODL controller in communicating the EAR-ABAC Policy-Engine machine, as shown in Fig. 8. The figure illustrates the network latency in communicating the EAR-ABAC Policy-Engine using different numbers of attributes and sending different concurrent requests from the SDN-ODL controller to the EAR-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)** finally using 24 attributes, the average time

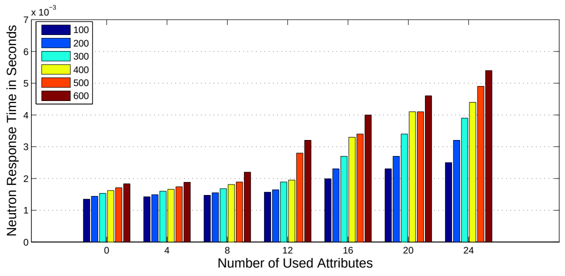

Fig. 7. The average time for Neutron communicating the EAR-ABAC Policy-Engine machine, including and excluding the user attributes.

for 600 concurrent requests is around 2.04 times of the time of 100 concurrent requests. While for zero attributes, the time for 600 concurrent requests is around 1.357 times of the time for 100 concurrent requests.

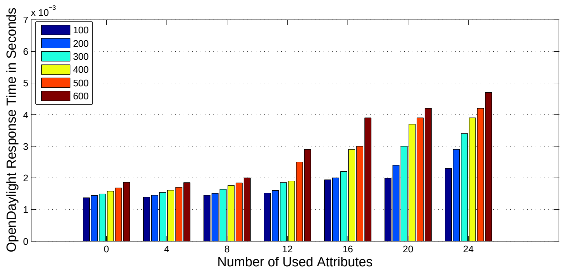

Fig. 8. The average time for SDN-ODL controller communicating the EAR-ABAC Policy-Engine machine, including and excluding the user attributes.

Finally the results indicate that the difference in average response time between the normal OpenStack (Zero attributes used) state and using the EAR-ABAC model attributes exist but it is too small. Also the difference in average response time when communicating the SDN-OpenDaylight is smaller than that when communicating the OpenStack Neutron service. So they can say that the EAR-ABAC can be used faster with the SDN-integrated cloud than when it is used with the normal cloud environment. This is related to the different nature between the OpenDaylight controller and the OpenStack-Neutron service.

4.3 Validation Analysis

In order to validate the extended model, the authors have to compare 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 the EAR-ABAC or other models can support, as shown in Table 2. Where each feature can be motivated and supported by EAR-ABAC or other models as follows:

- **Scalability:** The EAR-ABAC is scalable enough to deal with large numbers of users and also administrators. By using the role and task principles, which can classify the users to a set of roles each with it's own power to be able to access different objects based on the objects' sensitivity level. There is only an approach [11] which uses both principles, but it does not mention how administrator scalability can be ensured, and also is based on RBAC model. On the objects' side, EAR-ABAC is also scalable enough to deal with large numbers of objects, by classifying them based on the sensitivity levels (SL) function, that can classify the objects into different sensitivity levels.
- **Heterogeneity:** Since cloud services are delivered by a vast number of diverse technologies and mechanisms, which can cause heterogeneity threats [4]. Hence, heterogeneity in cloud computing can come as a result of differences at various levels, either software or hardware levels. Heterogeneity can also happen due to different types of mechanisms, domains and policies being used. The policy (Po) in EAR-ABAC that represents the attribute dominance relations, can cope with heterogeneity caused by security policies [15]. It should be noted that Sun et al. [18] used the ontology principle to deal with the heterogeneity issue.
- **Auditing:** To secure cloud computing and access control systems used in it. In access control systems, audit has to monitor a system's current state, record any failure to formulate a decision and report any attempt to violate the access policy or alteration of privileges. Moreover, it has to track and keep records about granted capabilities to subjects and any change applied to objects such as renaming, copying and deleting. the EAR-ABAC is based on the attribute-rules [15] for both users/ subjects and objects, Hence any change for any value of the subject's attributes will be recorded because it will change the subject capabilities. Also the same for the objects any change for any value of the object's attributes will be recorded, and change the object's sensitivity level. So the authors consider it as dynamic auditing.
- **Assign and ease of privileges:** Whenever, a small number of steps are required to assign or ease privileges, a number of mistakes can be reduced due to either human or system errors. In the extended model the role user assignment (RUA), task role assignment (TRA), and permission task assignment (PTA) are done internally and dynamically using other helping factors without the user or administrator interference, so it is very quick and the possibility of error seems to be zero.
- **Flexibility in attribute management:** Reaching an agreement about what kind of attributes should be used, and how many attributes should be taken into account for making access decision is a complex task [7]. AR-ABAC [15] overcomes this issue using the attribute-rules.
- **Policy management:** The extended model can support the policy (Po) that represents a set of partial ordering relations on all user and object attributes, called the attribute hierarchies or attribute dominance relations. It can regulate and organize the relations between users.
- **While:** Least of permissions (LoP), Delegation of capabilities (DoC), Separation of duties (SoD), and File syncing and sharing (FSS) are supported by the AR-ABAC list of constraints [15].

Also there are some important concerns for the SDN-integrated cloud computing access control model, that should be taken into account, such as:

- The service providers and users are likely to be in different security domains.
- The dynamic and random behaviors of users are a big concern and challenge for access control systems developers, as users have no time or location restrictions.

The extended model as shown in Fig. 1 and Fig. 2 can deal with the previous concerns by using the following concepts:

Table 2. The EAR-ABAC against the current access control models.

No.	Access Control Requirements	Current Access Control Models												
		MAC [2]	DAC [8]	RBAC [17]	R-BAC [3]	Wang et al. [21]	CoRBAC [19]	O-RBAC [20]	ARBAC [10]	T-RBAC [11]	Sun et al.[18]	ABAC [1]	AC3 [22]	EAR-ABAC
1.	Scalability	⋆	⋆	✓	●	✓	✓	✓	✓	✓	✓	●	✓	✓
2.	Heterogeneity	⋆	⋆	⋆	✓	⋆	⋆	⋆	⋆	✓	✓	⋆	✓	✓
3.	Auditing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4.	Assign and ease of privileges	⋆	⋆	⋆	⋆	✓	⋆	⋆	⋆	⋆	⋆	⋆	⋆	✓
5.	Flexibility in attribute management	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	✓
6.	Least of permissions	⋆	⋆	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7.	Delegation of capabilities	⋆	✓	⋆	⋆	⋆	✓	⋆	⋆	⋆	⋆	⋆	✓	✓
8.	Separation of duties	⋆	⋆	✓	●	✓	✓	✓	✓	✓	✓	✓	✓	✓
9.	File syncing and sharing	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	✓
10.	Policy management	⋆	⋆	⋆	✓	⋆	⋆	✓	⋆	⋆	⋆	⋆	✓	✓

✓ - Supported, ⋆ - Unsupported, and ● - Not applicable.

- Attribute-Rules, which can set an agreement about kind of attributes should be used, and number of attributes considered for making access decision, thorough sequence of steps [15].
- Task is another function used in AR-ABAC to restrict permissions and access assigned to roles. Where each user within the system is assigned a role; roles are given tasks that have permissions, and the task inherits it's power to access an object form the assigned role. Finally the task can access an object if and only if it's power dominates or equal the object's sensitivity level.
- Sensitivity levels are used in the extended model to classify the objects according to the possessed object attributes, object-rules, and environment conditions. Hence any task or process employed by a task need enough power to access objects, as there should be no access to any object without a power equal or dominate to the objects' sensitivity levels.

5. 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. [19]: 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 [20]: 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 [10]: 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 Giine [11]: 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. [18]: 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. [22]: 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.

6. CONCLUSION AND FUTURE EXTENSIONS

In this paper, the authors have extended the AR-ABAC access control model proposed by them earlier [15], to be used with the SDN-integrated cloud computing environments. The extended AR-ABAC is a new access control model for the SDN-integrated cloud computing environments, this model has a set of features that distinguish it from other traditional and current models proposed for cloud computing environments:

- The model can define an agreement on the attributes should be used and number of attributes should be taken into account for making access decisions;

- It provides four different ways to access the cloud IaaS;
- It has been integrated with the OpenStack Neutron service;
- The model has been integrated with the SDN-OpenDaylight controller and can work well with it;
- The EAR-ABAC provides the basic operations for the cloud root user and the tenant root user; and also have the ability to work with other types of controllers such as Floodlight [5].

The experimental results have shown that the EAR-ABAC is suitable for the SDN-integrated cloud IaaS. Where the difference in average response time between the normal OpenStack (Zero attributes used) state and using the EAR-ABAC model attributes exist but it is too small. Also the difference in average response time when communicating the SDN-OpenDaylight is smaller than that when communicating the OpenStack-Neutron service. So the authors can say that the extended EAR-ABAC can be used faster with the SDN-integrated cloud than when it is used with the normal cloud environment. This is related to the different nature between the OpenDaylight.
Finally the future extensions for this work can be in two directions. The first one is to integrate that model with the SDN-FloodLight [5] controller. Although the SDN-OpenDaylight and the SDN-FloodLight controllers both are Java-core, but there is big differences between them. The second direction is to try integrating this extended access control model with other OpenStack services such as: Horizon and Swift.

Acknowledgment

The authors would like to thank the National Natural Foundation of China for partly supporting this work (Grant No. 61472032 and 61170264).

7. REFERENCES

[1] M.A. 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.

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

[3] Achim D. Brucker, Lukas Brügger, Paul Kearney, and Burkhart 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.

[4] S. Crago, K. Dunn, P. Eads, L. Hochstein, Dong-In Kang, Mikyung Kang, D. Modium, K. Singh, Jinwoo Suh, and J. P. Walters. Heterogeneous cloud computing. In *2011 IEEE International Conference on: Cluster Computing (CLUSTER)*, pages 378–385, September 2011.

[5] Project Floodlight: Open Source Software for Building Software-Defined Networks. Available online: http://www.projectfloodlight.org/floodlight/. *(accessed on 6 December 2015).*

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

[7] Xin Jin, Ram Krishnan, and Ravi 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.

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

[9] Peter Mell and Timothy Grance. The nist definition of cloud computing. Special Publication 800-145, U.S. Department of Commerce, October 2012. National Institute of Standards and Technology.

[10] Ei Ei Mon and Thinn Thu 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.

[11] 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.

[12] OpenDaylight (ODL). Available online: http://www.opendaylight.org/. *(accessed on 6 December 2015).*

[13] Open Networking Foundation (ONF). Available online: https://www.opennetworking.org. *(accessed on 6 December 2015).*

[14] OpenStack. Available online: https://www.openstack.org/. *(accessed on 6 December 2015).*

[15] Khaled Riad, Zhu Yan, Hongxin Hu, and Gail-Joon Ahn. Ar-abac: A new attribute based access control model supporting attribute-rules for cloud computing. In *2015 IEEE International Conference on Collaboration and Internet Computing (CIC 2015)*, pages 28–35, October 2015.

[16] Pierangela Samarati and Sabrina Capitani 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.

[17] 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.

[18] Lili Sun, Hua Wang, Jianming Yong, and Guoxin 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.

[19] Zhu Tianyi, Liu Weidong, and Song 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.

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

[21] Wenhui Wang, Jing Han, Meina Song, and Xiaohui 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] Younis A. Younis, Kashif Kifayat, and Madjid Merabti. An access control model for cloud computing. *Journal of Information Security and Applications*, 19(1):45 – 60, 2014.