Access control for on-demand provisioned cloud infrastructure services
Ngo, C.T.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 1

Introduction

Cloud Computing is effectively used to improve scalability, availability, elasticity and security of IT management in many application areas. It adopts advantages of many technologies such as virtualization, service-oriented architecture, Grid Computing and Utility Computing to allow customers and providers to cut costs on system deployments and operations. Many studies and best practices documents related to clouds deployment, design, development, operations and management have been proposed to incorporate above technologies [1–6]. Clouds in such approaches enable users’ data to store on share virtualized cloud infrastructures, which are on-demand provisioned at providers’ facilities. The virtualized infrastructures capacities can be elastically scaled up and down depending on varying users’ demands. In clouds, the diversity of accesses on shared resources brings challenges to protect confidentiality, integrity and availability. The cloud systems must guarantee unauthorized parties cannot access or modify protected resources. Therefore, access control is one of the crucial component in the cloud security. In this thesis, we focus on designing flexible and efficient access control mechanisms to protect cloud resources and simultaneously inter-operate with cloud infrastructures of providers.

1.1 Cloud Computing Characteristics

Cloud Computing was presented as the prospective development model of distributed computing which would influence the whole IT industry. It was initially introduced as a new product on utility computing by Amazon Web Services (AWS) [7], along with different concepts and definitions [1, 3, 5, 8–10]. It has been developed to address scales of data and service processing in both scientific and industry communities with efforts to reduce consumers’ costs and optimizing resource utilization. Although there are many Cloud Computing definitions [1, 3, 5, 8], the most popular one is from US. National Institute of Standards and Technology (NIST) [5] including essential characteristics, deployment models and service models. Figure 1.1 presents the conceptual reference model of the NIST Cloud Computing architecture.
According to NIST [5], Cloud Computing has the following essential characteristics:

- **On-demand self-service:** consumers can obtain services as and when needed without requiring human interaction.

- **Broad network access:** Users can access services using a wide range of network clients such as mobile phones, tablets, laptops, and workstations.

- **Resource pooling:** the provider’s computing resources are not dedicated to particular consumers but are pooled to serve multiple consumers using a multi-tenant model.

- **Rapid elasticity:** the amount of allocated resources can be provisioned and expanded rapidly commensurate with consumers’ demands.

- **Measured service:** amount of resources consumed is measured by a metering capability that appropriates to the type of services, (e.g. storage, processing, bandwidth, and active accounts).

NIST also defines other Cloud Computing aspects such as service types, deployment models, use-cases, and business opportunities [5]. The classification of cloud services relates to the separation of responsibilities between providers and customers on system managements. Prior to the clouds, customers were responsible for their whole computing system stack, from hardware devices, operating systems to middleware, and applications. Normally they were all deployed on-premise at customers’ sites. Cloud Computing was then proposed to provide separations of responsibilities on layers, in which the lowers were essentially outsourced to
CLOUD COMPUTING CHARACTERISTICS

Figure 1.2: Scope of controls between provider and consumer in NIST cloud services [9]

providers. It also provided the scalability and cost reduction in deployment, operation and management. Depending on either cloud providers or customers are in charge of what parts, cloud services can be classified into the following types:

- Infrastructure as a Service (IaaS): providers take care the networking, storage, computing and virtualization platforms, while customers control operating systems, middleware, runtime libraries and applications. Examples of this cloud service type are the Amazon EC2 [11], Microsoft Azure Compute & Network Services [12], Google Compute Engine [13], and Rackspace Managed cloud [14].

- Platform as a Service (PaaS): In this model, operating systems, middleware and runtime libraries are managed by providers. The customers need to focus only on their deployed applications and data. Typical PaaS platforms are Google App Engine [15], Microsoft Azure [12], Salesforce Heroku [16].

- Software as a Service (SaaS): this type of service gives customers the whole application service as if it is running on-premise. The difference is that providers take care on management of all the system from hardware, operating systems, networking to application deployment. This service model can be seen in Google Apps [17], Apple iCloud [18] and Microsoft Office 365 [19].

The Figure 1.2 illustrates the separations of controls between providers and consumers in mentioned cloud service types.

These cloud services can be deployed in different deployment models [5]: (i) private cloud, (ii) community cloud, (iii) public cloud or (iv) hybrid cloud. Actors in clouds use-cases are therefore defined according to the respective cloud architecture [9], including cloud consumer, cloud provider, cloud auditor, cloud broker and cloud carrier.
1.2 Convergence of Cloud Infrastructures and Optical Network Virtualization

In the research efforts for developing advanced cloud infrastructures using future network technologies, GEYSERS project [10] developed concepts and solutions corresponding to the above Cloud Computing characteristics.

- Huge increase in the number of users/applications and a rapid increase in available bandwidth for users beyond 1Gbps: GEYSERS defined and developed a novel dynamic wavelength service provisioning mechanism that enables network operators to manage their capacity to support large number of users with high bandwidth optical connectivity.

- High bandwidth requirement applications with 10Gbps or more become more popular in transferring data between data centers, networks for HD and SHD multimedia content distribution, large remote sensor networks or huge scientific data. GEYSERS defined and implemented a new mechanism for network operators to request and setup scheduled high bandwidth optical network connectivity between endpoints in an on-demand manner.

- GEYSERS provided multi-domain, inter-provider network and IT resources to users. They are managed by the consistent, dynamically provisioned security and access control policies.

- For applications requiring large-scale convergence of IT and network services which are similar to Amazon virtualized services and Microsoft SharePoint, GEYSERS contained a novel end-to-end service provisioning mechanism that automatically and efficiently bundles suitable IT resources with the required optical network connectivity services to provide to the user in a single step in an on-demand manner.

- GEYSERS defined and developed methods allowing infrastructure providers to partition their resources (optical network or IT resources) to compose logical infrastructures and offer to network operators as a service. The logical composition mechanism supported dynamic and on-demand changes of combined optical network and IT resources.

The GEYSERS reference model is illustrated in Figure 1.3. The project was aiming to provide the coordination between optical networks and IT resources across multiple provider domains, including infrastructure providers and network operators. Based on the novel mechanism to partition infrastructure resources to compose and deployed at different providers, the architecture implies a new business framework with cost and energy efficiencies. Such features bring requirements on architecture and system design as well as implementation [20].

GEYSERS models entities participating to specific workflows defined in its use-cases [21]. Nowadays, telecom and IT service operators use integrated roles
of infrastructure providers and infrastructure operators. It is expected they own the physical infrastructure, operate it, and finally, run services on top of it. This approach is highly inflexible, inefficient and extremely expensive as a whole, but not in specific business sub-processes. In the new architecture, GEYSERS enhances the current business models with following roles to reflect the importance of virtualization of the network and IT infrastructure:

- **Physical Infrastructure Provider (PIP):** The PIP role in GEYSERS implements the possession and operation of the physical infrastructure.

- **Virtual Infrastructure Provider (VIP):** The VIP role implements the virtual resource handling and composition for producing Virtual Infrastructures (VIs).

- **Virtual Infrastructure Operator (VIO):** The VIO role implements the configuration and operation of the infrastructure and also provides final services to consumers.

The research presented in this thesis is carried out in the context of and supported by the GEYSERS project [10]. The proposed architecture encounters challenges on designing and implementation access control mechanisms for distributed, inter-domain, multi-provider environments. It motivated us to perform research on access control approaches for inter-domain and multi-provider cloud infrastructure systems. The following section defines necessary requirements for access control in cloud infrastructure systems.
1.3 Access Control Requirements for Clouds Service Providers

From Cloud Computing characteristics identified by NIST and in the context of multi-provider cloud infrastructures in GEYSERS, the access control for cloud infrastructure systems should contain following features:

- **On-demand provisioning and self-configuration**: cloud resources are normally allocated and adjusted dynamically according to customers' requirements. They are reflected by the on-demand updating of resource meta-data. The access control mechanisms to manage cloud resources should bind and synchronize to these dynamic meta-data.

- **Fine-grained access control**: The access control mechanism must support rule-based definitions which are flexible enough to adapt different access control use-cases. More specific, it should provide ways to care about rich conditions requirements (i.e., who, what, when, where, why and how clauses in authorization statements) to consent accesses.

- **Flexible multi-tenancy**: An essential characteristic of clouds is to serve pooled resources for multiple customers following the multi-tenant model. In this manner, the proposed access control system must provide the multi-tenancy features by design.

- **Scalability**: A cloud provider should be able to manage its virtualized resources to serve large scale of customers. It requires that system performance must be critical in adopting suitable access control mechanisms.

- **Distributed**: Business models and use-cases of cloud infrastructures target for distributed environments. Therefore we need to provide mechanisms to inter-operate access control systems among multiple distributed cloud providers.

In the next section, we revisit existing access control models and most popular policy languages to identify what can be used in the cloud security. They will be used to justify our research goals in section 1.5.

1.4 Related Work

1.4.1 Preliminaries on Access Control Models

Access control has the purpose to protect the confidentiality and integrity of the system information. Besides authentication systems are in charge of confirming the truth of users identities, the access control regulates subjects' operations on data and resources. Based on functionalities, an access control system can be separated into the decision and the enforcement components [22, 23]. In the Access Control Framework for Open Systems ISO 10181-3 [22] illustrated in Figure 1.4, the initiator is the active subject who submits access requests to the Access
Control Enforcement Function (AEF). Here they are mediated by sending decision requests to a Access Control Decision Function (ADF) which determines whether they should be granted or denied. Depending on responded decisions, the AEF will enforce requests, e.g. either subjects can access the resources or denied messages are thrown.

![Diagram of ISO 10181-3 access control framework](image)

Figure 1.4: ISO 10181-3 access control framework [22]

In [23], Policy Enforcement Points (PEPs) at resource locations send decision requests to the Policy Decision Point (PDP), which is in charge of giving decisions from the predefined policies. In this way, policies can be stored and managed separately from resources. Access control models therefore are proposed to define interactions between PDPs and PEPs, as well as manage policies in authorization systems. Although there are many different access control models for various systems, they can be classified into following types:

1.4.1.1 Discretionary Access Control

In the Discretionary Access Control (DAC), a resource is assigned the ownership to one or more entities. The owners have all controls to decide who can access the protected resources and which permissions they allow to do. Policies in DAC models are implemented by either the access control matrix or the access control list [24, 25].

1.4.1.2 Mandatory Access Control

The Mandatory Access Control (MAC) is designed to prevent illegitimate information leakage based on clearance definitions. The data owners cannot set permissions like in the DAC. Instead of that, resources and subjects are attached security labels. The system configuration defines access rules based on labels and enforce them strictly.

The typical access rules in MAC are security clearance levels: i.e., labels could be in partial order-sets: top-secret, secret, confidential, restricted, official, unclassified, clearance [25]; or in categories, in which different areas are disjoint and competence (e.g. the Chinese Wall model [26]). The best known model to secure information flows is Bell-LaPadula model [27] which uses both MAC and DAC.
1.4.1.3 Role-Based Access Control

The Role-based Access Control (RBAC) families [28–30] are introduced with roles as an abstraction layer decoupling users and permissions. Rather than assigning directly to users in DAC models, the basic RBAC₀ model groups permissions into roles according to task descriptions, which is known as the role engineering process. Users assigned to roles will contain all permissions of the active roles. In extensions, roles can be organized in hierarchy in RBAC₁ [29], constraints to limit user and role assignments with RBAC₂ and RBAC₃. In practical, RBAC approaches are applied in different databases and operating systems. The common feature of these systems is that they have stable structures in which tasks are defined static. Hence, role organizing after the role engineering process mostly is unchanged. Rather, RBAC systems have drawbacks to support fine-grained authorizations such as dynamical assigning roles according to variable contexts (e.g. time, location, authorization states), which may lead to role explosion problems [31, 32].

Some extended RBAC approaches are proposed to support multi-tenancy features for clouds [33–35]. However they either have drawbacks in dynamic on-demand supports, fine-grained authorization and scalability [33, 34] or lack of practical mechanisms [35].

1.4.1.4 Attribute-based Access Control

To overcome limitations of RBAC systems, Attribute-based Access Control (ABAC) model is identified with the central idea that access can be determined based on present attributes of objects, actions, subjects and environment in the authorization context. It means that the ABAC is more flexible and scalable for real-time environments than RBAC. However, managing large number of attributes in ABAC is a challenge. Also, the complexity of attributes criteria in rules and conflict resolutions may arise during applying ABAC in access control for large-scale systems like cloud. ABAC implementation like eXtensible Access Control Markup Language (XACML) [36] only defines a general ABAC policy language with fine-grained authorization capability. Making it applicable with features analyzed in Section 1.3 requires both extensions and practical implementation mechanisms. This is the purpose of this thesis.

The Table 1.1 summarizes access control models adapting requirements in section 1.3.

In ABAC, the flexible and fine-grained authorization capabilities depend on the expressiveness of the provided policy language. Mechanisms such as AWS Identity

---

**Table 1.1: Access control models comparisons**

<table>
<thead>
<tr>
<th></th>
<th>On-demand self-service</th>
<th>Fine-grained authorization</th>
<th>Multi-tenancy</th>
<th>Scalability</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MAC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RBAC families</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>ABAC</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

...
and Access Management (IAM) JSON-style authorization policy [37], XACML standards [36, 38] provide different flexible degrees. For simplicity purposes, Amazon JSON-style policy [37] is quite limited compared to XACML [36] such as policy hierarchy, combining algorithms, attribute comparisons, etc. In this thesis, we adopt XACML [36] to present our authorization rules and policies. However, policies with the high expressiveness also have their complexity. So to apply XACML successfully, we need to investigate approaches in our implementation. Next section gives a brief overview of XACML to identify its implementation challenges.

1.4.2 Access Control Policy Languages

1.4.2.1 Ponder

Ponder [39] is a declarative, object-oriented policy language designed for distributed object systems. It consists of five types of policies: the authorization policies could define positive or negative decisions upon matching of subject, target resource, action and the optional environment constraint; the filtering policies extend by associating an action with filter expressions, where input or output parameters are defined; the delegation policies are used to transfer access rights defined in an authorization policy temporarily; refrain policies bind to subjects to avoid operations in spite of permitted by authorization policies; finally, the obligation policies define tasks must be performed upon specific events occurs. Ponder provides the ability to organize policies in groups and roles in hierarchy to reflect organizational structures, so supporting RBAC features.

However, Ponder has some limitations when applying in Cloud Computing. First, the language does not mention how to process and evaluate policies. The Ponder Toolkit only contains a compiler to transform policies in to a system dependent language (e.g. Java code). Secondly, in distributed environment, policies can be defined by different authorities while Ponder does not define policy issuers, thus assumes that all policies are trusted. The policy repository must implement an enforcement mechanism to manage trusted policies. And finally, the Ponder implementation is obsoleted and no longer supported [40].

1.4.2.2 PERMIS

PrivilEge and Role Management Infrastructure Standards (PERMIS) [41] is a role-based access control infrastructure using X.509 attribute certificates [42] to store users’ role values. In this system, their integrity is protected by digital signatures of Attribute Authorities (AAs). Storing attributes in X.509 certificates is similar to the Security Assertion Markup Language (SAML) [43] with the XML-based structure. To implement RBAC concepts, PERMIS uses permission attributes and role attributes. RBAC operations are defined by a XML-based policy language containing: subject domain policy to define group of subjects, role hierarchy policy to define role hierarchical relationships, policy to define assignments from roles to a subject domain with related validity periods and delegation depths (a.k.a the role assignment in RBAC), action policy to specify actions binding with a target resource, and policy to assign actions to a role under conditions (a.k.a the RBAC
permission assignment). Policies are created by Privilege Allocators at AAs, then are stored in Lightweight Directory Access Protocol (LDAP) directory services. The system follows the ISO 10181-3 Access Control Framework [22]. In this framework, the ADF retrieves policies from LDAP services, along with attribute certificate revocation lists to validate principals’ attribute certificates.

In general, the XML-based policy language in PERMIS tried to implement RBAC operations with X.509 attribute certificates. Therefore, it suffered drawbacks from RBAC models with the stable role structure. The policy to assign actions to role containing conditions with Boolean logic could provide some levels of fine-grained authorization. However, policies in PERMIS have only two decisions: either permit or deny-by-default, so it cannot handle well intermediate results upon errors or undefined cases (i.e., conflicting decisions or undefined policies) as in the XACML.

1.4.2.3 XACML

XACML [36, 38] is an authorization policy language in XML format based on the ABAC model. It composes policies from set of attribute criteria joined by logical operators, which are used to decide answers for authorization requests. XACML is scalable in arrange policies in hierarchy order which can be combined and extended vertically by conflict resolution algorithms. In addition, by supporting delegations, obligations and advices, XACML is applicable in many areas such as networking, grids, clouds, enterprise organization and management. However, the growth of policies to address system scales will increase complexity of the policy repository, which leads to the drop of policies evaluation performance.

XACML v3.0 [36] organizes components following the model in Figure 1.5. In the model, policies are arranged hierarchically as follows:

- A policy-set is composed from a set of policies or other policy-sets.
- A policy can contain a set of rules.
- A rule has a logical expression of attribute values as the criteria for rule’s decision.
- Parent’s decisions of policy elements are the joined of children’s according to the predefined combining algorithms.

The XACML is considered as a de facto standard for the attribute-based authorization policy language. Adopting XACML allows systems to have advantages supporting complex fine-grained authorization scenarios than other policy language. In spite of that, due to complexities in its evaluation semantics, there are gaps to apply XACML standards for high performance systems.

The dynamic and elastic features in Cloud Computing identified in section 1.1 require a flexible access control model like the ABAC for multi-tenant systems. It requires a rich expressive policy language supporting sophisticated, fine-grained authorization rules. However, the implementation mechanism needs to have efficient and scalable performance to adapt large-scale cloud management systems at providers. In the next section, we identify and formulate research questions on how to resolve these challenges.
1.5 Research Questions

In this thesis, we focus on model designs and implementation techniques for access control solutions for cloud infrastructure services. They are investigated by the following research questions:

1. How do we design a flexible and scalable access control model supporting the on-demand provisioned self-service of cloud infrastructure services?

   In order to support large numbers of users with elastic resource scaling, cloud providers need to apply on-demand provisioning mechanisms to manage their resource capacity. As depicted in section 1.2, they are either the optical wavelength service provisioning for network operators, or the computing and storage share resource pool mechanism for IT service providers. In general, these mechanisms generate changes in numbers and properties of resources over fine-grained time resolutions, e.g. minutes or hours. However, existing access control models are eventually designed to manage a stable numbers of resources. Whenever there's a change in the system e.g. adding new or removing old resources, policies should be reconfigured, such as role engineerings in RBAC or updating attribute-based policies in ABAC. There is no access control model that is aware of life cycles of provisioned resources. We need a formal model along with its mechanisms binding between service life-cycles of cloud infrastructures and their access control policy management. This question will be investigated and answered in Chapter 2.

2. How do we design an access control model for cloud infrastructure providers
in which customers can manage their own virtualized resources distributing in multiple domains?

From cloud providers’ perspective, cloud systems should support sharing resource pools to serve multiple customers, where they are able to customize and exploit subscribed services in an isolated manner. It is known as the multi-tenancy [5, 44]. In this manner, the access control model for clouds providers should be designed with multi-tenant features. Tenants are able to customize policies for their services, while still under the scope of the provider’s policies. The NIST cloud definitions [5] said that cloud services are able to scale up extensively, not only from a single provider, but also from multiple, distributed providers. It is illustrated in the GEYSERS approach [10], in which partitioned resources at multiple providers can be aggregated to compose customers virtual infrastructures [21]. Therefore cloud providers should collaborate to offer inter-provider cloud infrastructures composed from resources crossing distributed domains. Such sophisticated systems are usually provisioned in multiple domains using service lifecycle managements [45, 46]. Designing the access control model for inter-provider systems should be aware of these characteristics. We solve this question in chapters 2 and 3.

3. How do we implement a high performance authorization policy evaluation engine, which should be required in access control solutions for cloud providers?

Mechanisms for access control models provide different expressiveness. They represent how much flexibility policies can be configured to moderate managed resources, i.e., defining criteria in clauses who, what, where, when, how of authorization statements. However, access control is eventually one of overheads of systems. Such mechanisms should be aware of the overall system performance of cloud providers, which should handle hundreds or thousands of customers at the same time. Normally, the more expressive the policy language is, the more overhead the access control adds to the system. In this thesis, we adopt XACML as the attribute-based policy language for fine-grained authorization purposes. The overhead when applying XACML motivates us to investigate and introduce the mechanism in chapters 4 and 5 to efficiently improve system performance.

1.6 Contributions

The contributions in this thesis are as follows:

- Chapter 2: We propose an access control model for multi-tenant cloud services using attribute-based policies. The model could integrate with the existing information model of cloud resource managements, which allows dynamic updates and reconfigurations regarding system scaling. It not only has scalability in terms of number of resources, but also allows delegations and collaborations among tenants in multiple levels.

- Chapter 3: In this section, we extend the proposed multi-tenant access control model for distributed multi-domain Intercloud infrastructures. It used the
proposed token exchange mechanisms among providers aiming to guarantee model's characteristics in the distributed environment. The extension is then validated in set of Intercloud scenarios involving multiple cloud providers.

- Chapter 4: XACML is a widely used attribute-based policy language that contains rich functionalities and expressiveness. To facilitate the adoption of this policy language in our access control model, we analyze and express the XACML logical model, then design new decision diagram data structures to represent its elements. These mechanisms could be applied to solve problems in different XACML policy managements, including the high performance policy evaluation engine in the next chapter.

- Chapter 5: using defined mechanisms in the previous section, we build up a high performance XACML policy evaluation engine. It not only has magnitudes of throughputs compared to previous work, but also maintains the same policy semantics and expressiveness [47]. The engine is used as the basis for building high performance PDPs in our multi-tenant access control model which is implemented and deployed for complex cloud infrastructure services.

Finally, Chapter 6 summarizes results presented in previous sections to answer the identified research questions.