Regular Paper

Risk Adaptive Authorization Mechanism (RAdAM) for Cloud Computing

Doudou Fall[1,a) Takeshi Okuda[1,b) Youki Kadobayashi[1,c) Suguru Yamaguchi[1,d)

Received: June 22, 2015, Accepted: December 7, 2015

Abstract: Cloud computing provides many advantages for both the cloud service provider and the clients. It is also infamous for being highly dynamic and for having numerous security issues. The dynamicity of cloud computing implies that dynamic security mechanisms are being employed to enforce its security, especially in regards to access decisions. However, this is surprisingly not the case. Static traditional authorization mechanisms are being used in cloud environments, leading to legitimate doubts on their ability to fulfill the security needs of the cloud. We propose a risk adaptive authorization mechanism (RAdAM) for a simple cloud deployment, collaboration in cloud computing and federation in cloud computing. We use a fuzzy inference system to demonstrate the practicability of RAdAM. We complement RAdAM with a Vulnerability Based Authorization Mechanism (VBAM) which is a real-time authorization model based on the average vulnerability scores of the objects present in the cloud. We demonstrated the usefulness of VBAM in a use case featuring OpenStack.

Keywords: cloud computing, risk, access control, fuzzy inference, xacml

1. Introduction

Modern computer systems are highly dynamic. This fact is epitomized by cloud computing, which is a novel paradigm in Information Technology (IT) that features data externalization and low cost of computing infrastructures. The National Institute of Standards and Technology (NIST) [1] defines cloud computing as "a model for enabling ubiquitous on-demand network access to a shared pool of configurable resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." Despite the fact that cloud computing is profitable for both adopters and providers, its security issues remain a concern; particularly, the fact that traditional static authorization mechanisms are employed to enforce access decisions. The current situation reflects an unsuitability between cloud computing and the authorization mechanisms to date. Novel authorization mechanisms are needed. In the early 2000s, the JASON report [2] pioneered the usage of risk in modern authorization mechanisms in order to keep with the dynamicity of modern IT platforms.

Their idea supposes that every access request is associated with a risk decision that will be challenged by a threshold to allow (resp. deny) the access request. That marks the genesis of a number of work about risk-aware access controls. Among those researches, a seminal paper [3] defined the principles of a risk aware access control. In that regard, we propose a risk adaptive authorization mechanism (RAdAM) which is a dynamic real-time risk-aware authorization mechanism for cloud computing. We ascertained that RAdAM follows the principles of a risk aware access control as defined in Ref. [3]. We proposed authorization mechanisms for the most prominent cases that exist in cloud computing. We consider the situation where the user is a simple cloud customer with basic services. Afterwards, we slightly complicated the atmosphere by proposing an authorization mechanism for collaboration in cloud computing. Finally, we designed an authorization mechanism for the most complicated case in cloud computing: cloud federation. We employed a fuzzy inference system (FIS) [4] to demonstrate how RAdAM functions.

The main contribution of RAdAM is not in the risk determination itself but in the way the authorization algorithms work and the novel parameters we introduce in order to allow the authorization to be as flexible as possible. To complement RAdAM, we also proposed a vulnerability-based authorization (VBAM) mechanism which is a variation of the Bell-Lapadula multi-level security (MLS) [5]. The most important point of VBAM is the usage of the vulnerability score of the objects to enforce a decision. We contend that modern authorization mechanisms should not only be dynamic, but also should be tailored to the vulnerabilities of the objects that are being accessed. We demonstrated the usefulness of VBAM by explaining how it could be used in an OpenStack environment. The rest of the paper is structured as follows: Section 2 contains the motivation and the background explanation of a risk aware access control. The related work is developed in Section 3. We explain the details of RAdAM in Section 4. We expand upon the usage of RAdAM in a fuzzy inference system in Section 5. Section 6 contains the explanation of VBAM and a use case of a possible usage. In Section 7, we

[1] Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a) doudou-f@is.naist.jp
[b) okuda@is.naist.jp
[c) youki-k@is.naist.jp
[d) suguru@is.naist.jp

discuss the shortcomings of our proposals and future work that lies ahead. Section 8 concludes the paper.

2. Background: Risk Aware Access Control

Information sharing has always been a subject of controversy. That is why it drew the attention of the security experts in the early stages of computing. Many security mechanisms have been developed for the purpose of information sharing: starting from the basic Access Control Lists (ACLs) to Mandatory Access Control (MAC) [6], Discretionary Access Control (DAC) [7], and Role-Based Access Control (RBAC) [8]. Currently, we are facing new challenges as the shared infrastructure became dynamic and those aforementioned ACs are not flexible enough to meet the new challenges. It became an urgent matter to engineer access controls that can cope with the new challenges brought by the modern computer systems. The JASON report [2] pioneered the idea of a modern access control by defining three fundamental principles that should be included in their design:

- Measure risk: "if you can measure it, you can manage it". In other words, knowing the risk associated with an event permits a better handling.
- Establish an acceptable risk level
- Ensure that the information is tailored to the level of the acceptable risk.

An authorization mechanism that is engineered by mirroring these guidelines can deal with the reality of today's information sharing platforms. A seminal paper about an access control that follows theses guideline was proposed by NIST and titled Risk-Adaptable Access Control (RAdAC) [3]. RAdAC is a real-time, adaptable risk-aware access control built by a combination of Attribute-Based Access Control [9], Policy-Based Access Control, Machine Learning, and heuristics. Six factors are indispensable to make RAdAC decisions:

- **Operational need**: this factor is one of the factors that RAdAC borrowed from traditional access controls. It involves the notion of *need to know*, which means that a requester should have a relation with the object he is requesting.
- **Security risk**: is the cornerstone of RAdAC. This factor requires the use of machine learning for a real time probabilistic determination of risk associated to a request.
- **Situational factors**: sometimes the access decisions are made depending on the actual situations; it is possible that the *Operational need* outweighs the *Security risk*.
- **Access Control Policy**: as any normal AC, policies need to be enforced. The policies should respect the mechanisms of RAdAC by, among other things, setting the acceptable risk level and defining the conditions on which the *Operational Need* can outweigh the *Security Risk*.
- **Heuristics**: the goal is to use past access control decisions to make present and future decisions. The utilization of past decisions will help to better determine the *Security Risk* and *Operational Need* and will increase the positive number of access control decisions.

3. Related Work

To the best of our knowledge, this technique is novel as it has not been employed by other researchers. Nevertheless, a quick perusal over the literature of risk based access control revealed interesting facts.

Despite the fact that cloud computing is the archetypical dynamic system, there is not a tremendous amount of research related to risk aware access control in cloud computing. In fact, Fall et al. [10] and Dos Santos et al. [11], [12] are the only researchers who are evangelizing the usage of this method in cloud computing. Fall et al. [10] proposed a semantic for adapting RAdAC [3] in cloud computing. Unfortunately they did not propose any implementation of their idea. Santos et al. [11] proposed a risk-based access control architecture for a highly scalable cloud federation. They demonstrated the usefulness of their proposal through a prototype by using a combination of tools including XACML [13]. In their other paper [12], they extended [11] by adding three modules to the XACML 3.0 architecture. A risk policy module that explains to the cloud service provider how the access control must be handled. The risk engine module handles the processing of risk policies. The risk quantification and web services module quantifies the risk for every access request.

Cheng et al. [14] extended the Bell-Lapadula multi-level security (MLS) access control model with the concept of risk and named their proposal FuzzyMLS. In FuzzyMLS, risk is the value of utility loss, multiplied by its leakage probability. The leakage probability is jointly dependent on the temptation and inadvertent leakage probabilities. The temptation is calculated by the security level of the subject and object, and the inadvertent leakage by a fuzzy approach. They use different risk thresholds to make access decisions. Ni et al. [15] proposed a risk based access control built on fuzzy inference. They claim that the inflexibility of traditional access control is a major inhibitor for information sharing. They first proposed fuzzy BLP and compared it to FuzzyMLS [14]. Moreover, they argue that despite fuzzy inference being an excellent solution for modern access control, it raises new issues that have to be addressed. Particularly, the fact that, generally, risk aware access controls are time-consuming thus malicious users can take advantage of the time window that exists between an access request and the risk mitigation. Further, they proposed some algebraic adjustment to solve the issues of conjunction and disjunction in fuzzy inference.

Burnett et al. [16] proposed TRAAC a trust and risk aware access control that provides a policy coverage, dynamic access control decisions, appropriate denial of risk and rights delegation. They claim that their system fits the healthcare domain perfectly though it can be extended to other domains. First, they defined a zone policy model where the data owner has total control of the privileges on his data which he can share with other users, hence including them in the domain. The trust is defined in terms of sharing trust and obligation trust that permits the verification of whether the requester respected the obligations that are assigned to him or not. They used a probabilistic computational trust model called *Subjective Logic* to formulate their trust assessment. The risk is computed in the classical manner of expected loss in terms of unwanted disclosure. Khambhammettu et al. [17] designed a risk aware framework for decision making where risk is evaluated as the product of threats and impact scores. The authorization is made by weighing the security risk against the operational-need and security factors. The framework embraces four approaches based on the sensitivity level of the object, the trustworthiness of the subject, and the remaining are made of the combination of the two aforementioned approaches. Throughout their paper, they proposed successful instances of their proposal. Shaikh et al. [18] proposed two methods of risk aware access control based on trust and risk. The formulation of trust and risk depends on rewards and penalties which are updated either positively (rewards) or negatively (penalties) after an access request decision is made (allow or deny, respectively). An increase of the trust causes a decrease of the risk and vice versa. The method is based on Multi-Level Security (MLS) systems [5]. Due to the slow responses in access rights, they proposed another risk aware access control based on Exponentially Weighted Moving Average (EWMA) [19]. This method has similar functionalities as the first method.

4. Risk-Adaptive Authorization Mechanism: RAdAM

In this section, we will shed some light on our proposal. We propose an authorization mechanism for three cloud deployment methods: user-mode, collaboration mode and federation mode.

Our authorization mechanism respects the aforementioned principles of RAdAC. We reiterate that, in this model, every access request is evaluated by a risk that will be challenged against a threshold. We introduce a number of parameters that are handy while designing the authorizations. If the access request is allowed (resp. denied), the parameters are ‘positively’ (resp. negatively) updated in a way that risk will be better (resp. worse) for the next access request. We assume the existence of a set of objects O and a set of subjects S. An object represents a piece of data which is controlled by the system.

4.1 RAdAM: Cloud User

We are aware of the fact that in basic cloud environments, the prominent actors are the users and the cloud administrator. We suppose that the cloud administrator is honest so our authorization mechanism focuses on the user. We interchangeably say user or subject. We consider the data of the user as objects. Thus in our authorization mechanism, we have the tuple <Subject, Object>. As we strive to build a real time risk adaptive authorization mechanism, we introduce two tokens that will be used to compute the risk associated with a request:

- α: represents the dynamic token of the user that is linked with the object the subject is trying to access.
- β: stands for the token of the object that is being accessed.

In our system, every single access is associated with a risk calculation. The risk is calculated in function of the token of the user α and the token of the object β. Risk is represented by $Risk(\alpha, \beta)$. To be in line with the principles of a risk adaptive authorization mechanism, we introduce a threshold $T(\alpha, \beta)$. The threshold is also inherently attached to the subject and that particular object. For *situational factor* purposes, we also introduce two other parameters:

- The average risk, which is the sum of the risk accumulated by the subject divided by the number of objects:
  $Risk(\alpha, \beta)_A = \sum Risk(\alpha, \beta)$ / n.
- The average risk threshold, which is the sum of the thresholds to all subject/object risk attempts divided by the number of objects:
  $T(\alpha, \beta)_A = \sum T(\alpha, \beta)$ / n.

The average risk gives us the general behavior of the user. In case the $Risk(\alpha, \beta)$ is equal to the threshold $T(\alpha, \beta)$, the system verifies whether the average risk is greater than or equal to the average risk threshold to subsequently allow the access request or deny it otherwise.

We want to clarify how the system works in detail as the parameters defined above are for generic purposes. We consider that we have a set of subjects S. A subject $s_i \in S$ has a set of objects $O_i = \{o_{i1}, \ldots, o_{in}\}$. So for an access request to object o_{ik}, the risk is defined by $Risk(\alpha_{ik}, \beta_{ik})$. Similarly the threshold is $T(\alpha_{ik}, \beta_{ik})$. The average risk would be given by:
$Risk(\alpha_i, \beta_i)_A = \sum_{j=1}^{n} Risk(\alpha_{ij}, \beta_{ij})$ / n. We used α_{ij} and β_{ij} in the summation where i represents the user thus constant and j represent the object and is variable. α_{ij} is the dynamic token of the user s_i who has a set of objects O_i so only j varies in the summation. β_{ij} represents the token of the object that is being accessed. The user s_i is constant and j varies according to the objects.
The average risk threshold is: $T(\alpha_i, \beta_i)_A = \sum_{j=1}^{n} T(\alpha_{ij}, \beta_{ij})$ / n. The authorization algorithm is highlighted in Eq. (1).

$$Au(s_i, o_{ik}) = \begin{cases} \text{if } Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) & \text{allow} \\ \text{if } Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ then, if} & \\ Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A & \text{allow} \\ otherwise, \text{ deny} & \end{cases} \quad (1)$$

4.2 RAdAM: Cloud Collaboration

4.2.1 Data Owner

In our understanding, collaboration in cloud computing reflects the idea of a subject expressing a desire to share their data with other subjects. While collaboration creates added values, resource misuse and irresponsibility inflict costs and damages on affected resources. Permission misuse and legitimate user attacks are among the most serious threats which users in cloud computing collaboration face today. The situation is made even worse due to the dynamic aspect of cloud computing. The authorization mechanism that we display in this section solely concerns the data owner. We discuss the authorization mechanism for the other subjects in the appropriate section. The parameters we defined in the previous section are carried over in this section. We also introduce new parameters that are inherent to collaboration in cloud computing:

- γ: represents the dynamic collaboration token of the user that is attached to the object which the user is trying to access.
- δ: represents the token of the object in collaboration mode.
- $Risk(\gamma, \delta)$: is the risk associated with a collaboration access

request.

- $T(\gamma, \delta)$: the risk threshold to which the risk will be evaluated against.
- $Risk(\gamma, \delta)_A$: the average collaboration risk is the sum of the accumulated collaboration risk divided by their number: $Risk(\gamma, \delta)_A = \sum Risk(\gamma, \delta)$ / n.
- $T(\gamma, \delta)_A$: The average collaboration risk threshold is the sum of the risk collaboration thresholds divided by their number: $T(\gamma, \delta)_A = \sum T(\gamma, \delta)$ / n.

The above-introduced parameters are solely used in the case of collaboration. Similar to the previous section, $Risk(\gamma, \delta)_A$ and $T(\gamma, \delta)_A$ are only used for situational factors.

For an instance of the authorization mechanism in this collaboration case, we still consider the user s_i with his set of object O_i={o_{i1},...,o_{in}}. If the user wants to access the object o_{ik}, the collaboration risk is given by: $Risk(\gamma_{ik}, \delta_{ik})$. γ_{ik} represents the dynamic collaboration token of the user s_i that is attached to the object o_{ik} that the user is trying to get access to. δ_{ik} represents the token of the object o_{ik} in collaboration mode. The collaboration threshold is: $T(\gamma_{ik}, \delta_{ik})$. The average collabration risk is determined by: $Risk(\gamma_i, \delta_i)_A = \sum_{l=1}^{t} Risk(\gamma_{il}, \delta_{il})$ / t; t being the number of objects, of the user, that are in collaboration mode. The average collaboration risk threshold is: $T(\gamma_i, \delta_i)_A = \sum_{l=1}^{t} T(\gamma_{il}, \delta_{il})$ / t. The authorization mechanism is displayed in Eq. (2).

4.2.2 Collaborative User

In this situation we assume that the data owner wants to collaborate with another user in the cloud and thus share their data with them. As there are different levels of permission (i.e, read, write or execute), we introduce risk bands that define the level of permission to which the user belongs to.

When a user attempts an access request, the system checks first whether he/she has the adequate collaboration parameters. The system then proceeds and computes the access risk based on the parameters. The result will determine to which risk band the user belongs to, hence his/her level of permission over the data he/she is trying to get access.

$$Au(s_i, o_{ik}, C) = \begin{cases} \text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \end{cases} & allow \\ \text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ then,} \\ \text{if } Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \quad allow \end{cases} \\ \text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ then,} \\ \text{if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \end{cases} allow \end{cases} \\ \text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ then,} \\ \text{if } Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \quad allow \end{cases} \\ otherwise, deny \end{cases} \quad (2)$$

4.3 RAdAM: Cloud Federation

4.3.1 Data Owner

Cloud federation is understood as indicating subjects from different cloud providers being able to access each other's data [20], [21], [22]. Cloud computing federation should cope with a heterogeneous environment and dynamic sets of users and access requests, and are under multiple administrative domains. There is a great deal of identity management that is involved in this paradigm and technologies like OpenID [23] appear to be quintessential in these settings. In our authorization mechanism, we ignore all the noise caused by the federation and only focus on the user who is sharing their data with another user from a different cloud. We believe that cloud federation incorporates both the simple cloud and the collaboration. Therefore, the parameters defined in the previous section are carried over to this section. To those parameters, we add the following:

- ϵ: represents the dynamic federation token for the user and the object he is trying to access.
- η: represents the token of the object that user is trying to access.
- $Risk(\epsilon, \eta)$: represents the risk of the federation request access.
- $T(\epsilon, \eta)$: represents the threshold of the federation risk access to which the federation risk will be compared to.
- $Risk(\epsilon, \eta)_A$: stands for the average federation risk and is calculated by dividing the sum of all the federation risks by the number of objects: $Risk(\epsilon, \eta)_A = \sum Risk(\epsilon, \eta)$ / n.
- $T(\epsilon, \eta)_A$: represents the average federation threshold risk and is computed by dividing the sum of all the federation risk thresholds by the number of objects: $T(\epsilon, \eta)_A = \sum T(\epsilon, \eta)$ / n.

The access request is allowed (respectively denied) if the risk for federation is strictly greater than (respectively, strictly lower than) the risk threshold for federation. The other parameters are used in the *situational factor*.

For an instance of the authorization mechanism in this federation case, we still consider the user s_i with his set of object O_i={o_{i1},...,o_{in}}. If the user wants to access the object o_{ik}, the risk federation is given by: $Risk(\epsilon_{ik}, \eta_{ik})$. ϵ_{ik} represents the dynamic federation token for the user s_i that is trying to get access to the object o_{ik}. η_{ik} represents the token of the object o_{ik} that the user s_i is trying to access. The federation threshold is: $T(\epsilon_{ik}, \eta_{ik})$. The average federation risk is determined by: $Risk(\epsilon_i, \eta_i)_A = \sum_{m=1}^{z} Risk(\epsilon_{im}, \eta_{im})$ / z; z being the number of objects that are in federation mode. The average federation risk threshold is: $T(\epsilon_i, \eta_i)_A = \sum_{m=1}^{z} T(\epsilon_{im}, \eta_{im})$ / z. The authorization mechanism is represented in Eq. (3).

4.3.2 Federative User

Cloud federation can be described as a cloud collaboration on a wider scale. In fact, Tang et al. [24] designed an RBAC model for collaboration cloud services but exclusively used cloud federation, throughout their paper, to test their solution. In our system, a data owner in cloud federation has the right to allow a user from another cloud (federative user) to have access to their data. In this case, the data owner decides the risk level (user token and object token) of the federative user and can, at any moment, revoke their rights. The authorization mechanism for the federative user works as in Eq. (2).

$$
Au = \begin{cases}
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) > T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \end{cases} \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) > T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) > T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} & \mathit{allow} \\
\text{if } \begin{cases} Risk(\alpha_{ik},\beta_{ik}) = T(\alpha_{ik},\beta_{ik}) \text{ and,} \\ Risk(\gamma_{ik},\delta_{ik}) = T(\gamma_{ik},\delta_{ik}) \text{ and,} \\ Risk(\epsilon_{ik},\eta_{ik}) = T(\epsilon_{ik},\eta_{ik}) \\ \text{then, if } \begin{cases} Risk(\alpha_i,\beta_i)_A \geq T(\alpha_i,\beta_i)_A \text{ and,} \\ Risk(\gamma_i,\delta_i)_A \geq T(\gamma_i,\delta_i)_A \text{ and,} \\ Risk(\epsilon_i,\eta_i)_A \geq T(\epsilon_i,\eta_i)_A \end{cases} \end{cases} & \mathit{allow} \\
\text{deny otherwise}
\end{cases} \tag{3}
$$

4.4 Outsourcing Data without Outsourcing Control

It is common knowledge that cloud computing prospective adopters are not comfortable knowing that the cloud service provider can access their data at anytime. This explains why many financial institutions prefer to build their own private cloud instead of using the public clouds. In this Section, we show that our model can be used to limit the power of the cloud administrator over the user's data. In a perfect world, the cloud admin should be limited to managerial rights over the user's data. We can use the risk bands as in collaboration to allow the user to have more power over his data that the cloud provider. But the concern that lingers is that the cloud admin administers the cloud without being able to have a minimal access to the users data.

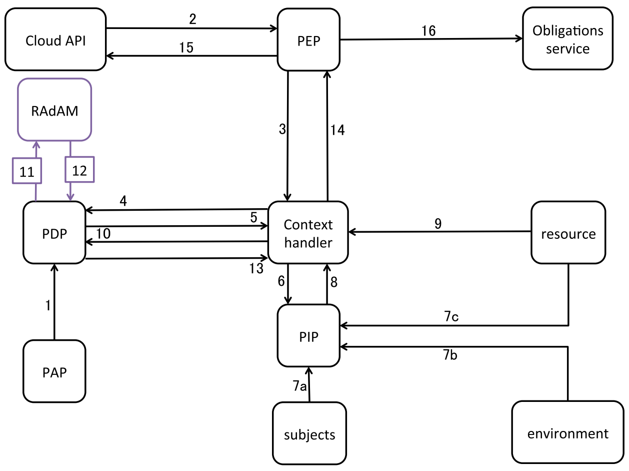

Fig. 1 RAdAM/XACML architecture.

4.5 Architecture of RAdAM

We use XACML 3.0 to leverage the architecture for our authorization mechanism which is depicted in **Fig. 1**. XACML is an extensible, XML-encoded language that provides format for authorization policies and access control decision request and responses. XACML 3.0 was approved as an OASIS standard on 22 January 2013 [13]. It includes a non-normative data flow model that describes the major components involved in processing access requests. Prior to any access request, the **policy administration point** (PAP) writes *policies* and *policy sets*, that represent the complete *policy* for a specified *target*, and transmits them to the **policy decision point** (PDP) (step 1). Via the cloud API, the user performs an access request (step 2). The **policy encryption point** (PEP) intercepts the request and forwards it to the **context handler** (step 3). The context handler converts the request into an XACML request contest and sends it to the PDP (step 4). The PDP requests to the context handler any supplemental attributes it might deem necessary to correctly evaluate the XACML request (step 5). The context handler requests the attributes from a **policy information point** (PIP) (step 6). The PIP retrieves the requested attributes which include, among other things, all the parameters that we introduced in our authorization mechanism and then transmits them to the context handler (steps 7–8). The context handler forwards the attributes to the PDP, and can optionally include the **resource** (steps 9–10). The PDP sends the attributes to the RAdAM module for evaluation (step 11). After evaluation as per of our authorization mechanism, the RAdAM module returns the decision to the PDP, which forwards it to the context handler (steps 12–13). The context handler translates the response to the native response format of the PEP and then sends it to the latter (step 14). The PEP transmits the response to the cloud API and fulfills the obligations, which include updating the tokens of the users (steps 15–16).

5. RAdAM Based on Fuzzy Inference System

Fuzzy inference systems (FIS) derive from fuzzy logic. Professor L.A. Zadeth of the University of California Berkley invented fuzzy logic in 1965 [4]. Fuzzy logic is a multivalued logic that permits intermediate values to be expressed in conventional evaluations like true/false, yes/no, high/low, etc. Fuzzy systems are an alternative to traditional notions of set membership and logic. The very basic notion of fuzzy systems is a fuzzy set. The concept of fuzzy set extends the concept of a classical crisp set. A classical crisp set is a collection of objects in a given range with a sharp boundary, which means that a member either belongs to that set or does not. The fuzzy set is fundamentally a more flexible set as it allows its members to have a smooth boundary i.e., a member can belong to a set to some partial degree. The fuzzy set convincingly provides interpretations that are similar to a human being thought processes. The implementation of fuzzy logic requires the three following steps:

- *Fuzzification*: the first step to apply a fuzzy inference system. It involves two processes: derive the membership functions for input and output variables and represent them with linguistic variables. This process is equivalent to converting or mapping classical set to fuzzy set to varying degrees. The membership function is a graphical representation of the magnitude of the participation of each input. It associates a weight with each of the inputs that are processed, defines functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. In practice, membership functions can have a multitude of different types, such as the triangular waveform, trapezoidal waveform, Gaussian waveform, bell-shaped waveform, sigmoidal waveform and S-curve waveform. The exact type depends on the actual applications. A triangular or trapezoidal waveform should be utilized for systems that necessitate significant dynamic variations in a short period of time. Systems that require a very high control accuracy are better suited with a Gaussian or S-curve waveform.
- *Fuzzy control rules*: The fuzzy classifiers represent one application of fuzzy theory. Expert knowledge is used and can be expressed in a very human-like way using linguistic variables, which are described by fuzzy sets. The expert knowledge can be formulated as rules:
  IF A is X **AND** B is Y **THEN** C is Z
  Linguistic rules describing the control system consist of two parts; an antecedent block (between the **IF** and **THEN**) and a consequent block (following **THEN**).
- *Defuzzification*: The two steps explained above constitute the fuzzy inference system. The fuzzy conclusion (output) is a linguistic variable and needs to be reverted back to the crisp value: the process is denominated defuzzification.

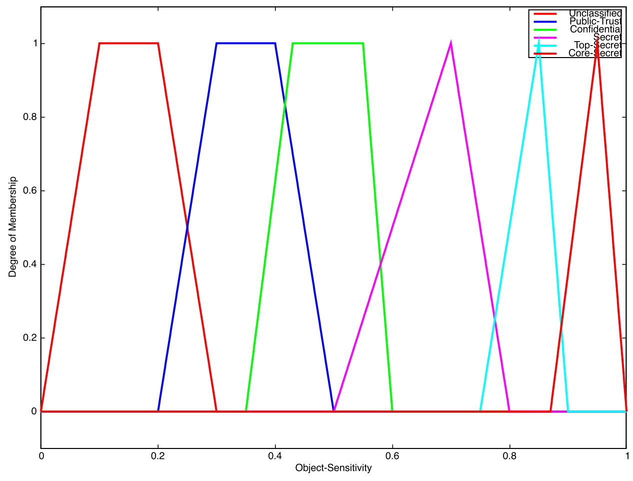

Fig. 2 Membership functions of the object sensitivty.

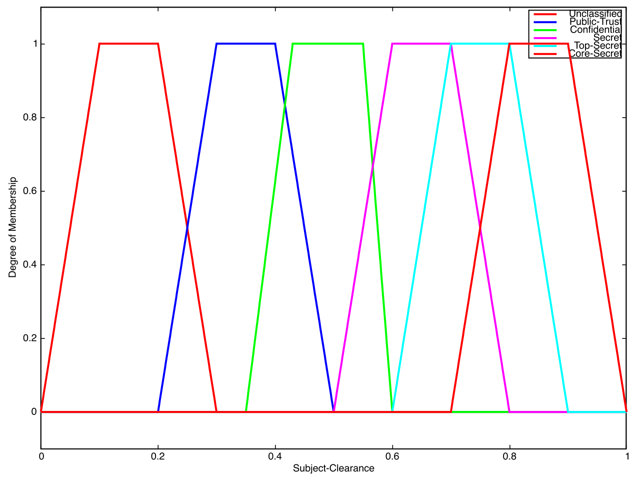

Fig. 3 Membership functions of the subject clearance.

5.1 RAdAM Risk Estimation in FIS

In order to explain RAdAM risk estimation in FIS, it is helpful to consider the following example. Let's suppose that we have a tuple <subject, object> in a cloud environment. The objects are defined by their sensitivity: unclassified (U), public trust (PT), confidential (C), secret (S), top secret (TS), and core secret (CS). Similarly, the different clearance levels for the subjects are as follows: unclassified (U), public trust (PT), confidential (C), secret (S), top secret (TS), and core secret (CS). As stated in previous sections, the risk is calculated in function of the subject clearance and object sensitivity and can be valued as negligible (N), low (L), medium (M), high (H), and very high (VH). The first step of the process of FIS consists of defining the input and output variables of our system. The inputs are the object sensitivity and the subject clearance and the output is the risk. We have decided to represent the clearance of the user in a trapezoidal membership function and the object sensitivity in a combination of trapezoidal and triangular membership functions. The risk is represented as a Gaussian membership function. The different membership functions are depicted in **Fig. 2**, **Fig. 3**, **Fig. 4**. We subsequently establish the fuzzy classifier (if ... then), which is shown in **Table 1**. Different values of the risk are evaluated in **Fig. 5** in function of the subject clearance and the object sensitivity.

Table 1 List of the rules for RAdAM risk evaluation.

S/O	U	PT	C	S	TS	CS
U	L	M	H	VH	VH	VH
PT	N	L	H	H	VH	VH
C	N	L	L	H	H	VH
S	N	L	L	L	H	VH
TS	N	N	N	M	L	H
CS	N	N	N	L	L	M

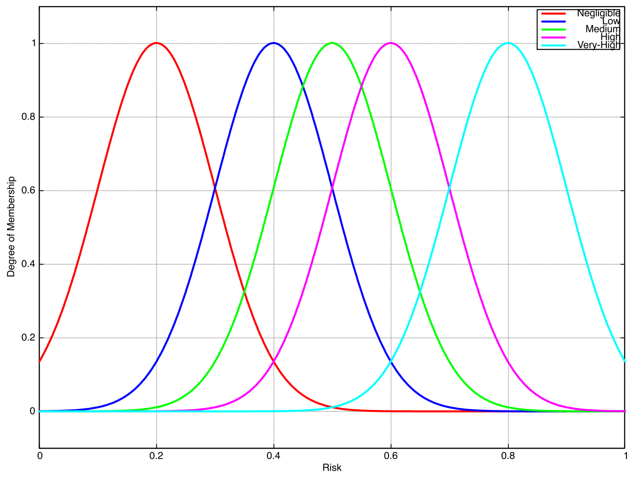

Fig. 4 Membership functions of the risk level.

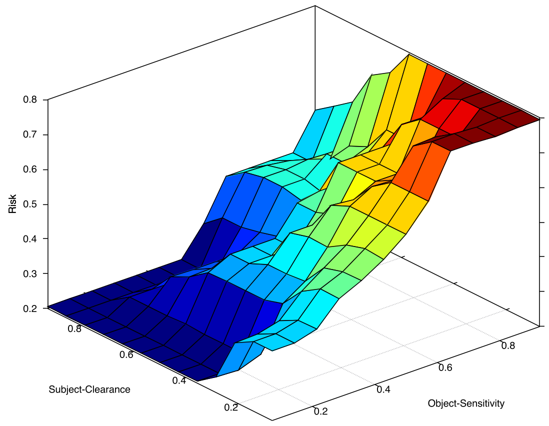

Fig. 5 RAdAM risk evaluation.

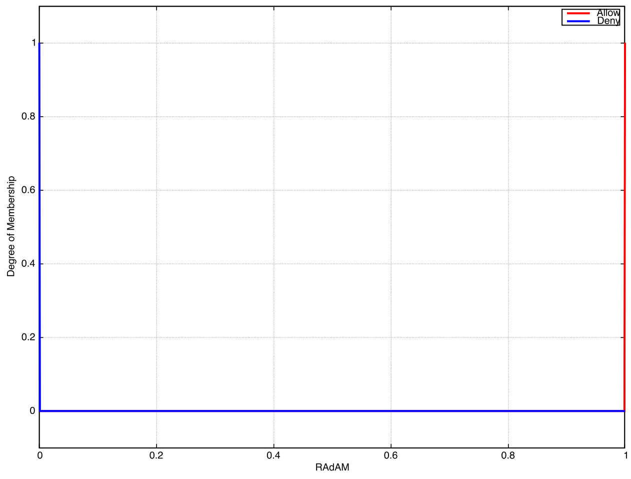

Fig. 6 Membership functions of RAdAM decisions.

5.2 RAdAM Decision Evaluation in FIS

The same process is conducted in order to evaluate the decisions of our authorization mechanism. It is important to note that

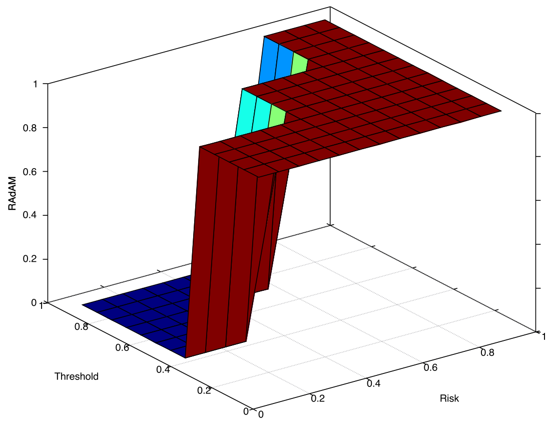

Fig. 7 RAdAM decision evaluation.

the threshold is also a risk. In this fuzzy system, the risk and the threshold are the inputs and the decision (allow or deny) is the output. The output is considered a constant as represented in **Fig. 6**. The schema of the decision results is depicted in **Fig. 7**. We can clearly see the decisions depending of the values of the risk and the threshold.

6. Enforcement of RAdAM with Vulnerability Based Authorization (VBAM) Mechanism

Until now, researchers in the access control fields have not thought of using vulnerabilities score to determine access decisions. We propose such solution by introducing Vulnerability-Based Access Control. The modus operandi is to tailor an access to an object to the vulnerability level of both the requester and the object that is being requested. In traditional access control, users are granted access to the objects regarding the privileges (read, write, or execute). We contend that these types of access controls are obsolete in modern computer systems like cloud computing. For instance, let us say we have a user who has the privilege to read an object. In simplistic thinking we might think that the user has a low privilege over the object because he can only read it. But the reality is that, if the object has a vulnerability with a high impact, the user, if malicious, can exploit the vulnerability and damage the system. That is the reason why we are advocating the usage of a Vulnerability (Risk) Based Authorization Mechanism (VBAM) where the users access to an object will be decided by the security level of the user and the average vulnerability score of the object that is being accessed. In our model, there is a need to have a standard vulnerability framework. The objects are classified according to their vulnerability score: risk level 1, risk level 2, . . . , risk level n. The users of the system are also classified according to the same multi-level security system. Whenever a user attempts an access, the system checks the security clearance of the user before computing the average vulnerability score of the object that is being accessed. If the user has a clearance level higher or equal to the clearance level of the object then the access is granted, otherwise it is denied. In this situation, if a user can only access objects with a low risk level, and if they succeed to exploit the vulnerability, then the damage would not be significant.

6.1 Practical: VBAM with the Common Vulnerability Scoring System

The basic premise of the traditional MLS Bell Lapadula model is to determine if a subject is trustworthy enough and has the legitimate *need-to-know* to access an object. A subject is usually a person or an application running on behalf of a person. An object is usually a piece of information such as a file. Each subject or object is tagged with a *security label* which represents the sensitivity level. A subject's sensitivity level reflects the degree of trust placed on the subject. An object sensitivity level indicates how sensitive the object is or the magnitude of the damage incurred by an unauthorized disclosure of the object. A subject can *read* an object if their label dominates the object's label

6.2 NVD and CVSS

The National Vulnerability Database (NVD) [25] is a publicly available database for computer-related vulnerabilities. It is the property of the United States (US) government, which manages it throughout the computer security division of the U.S. National institute of Science and Technology (NIST). The NVD is also used by the U.S. government as a content repository for the Security Content Automation Protocol (SCAP). The primary sources of the NVD are as follows: Vulnerability Search Engine (Common Vulnerability Exposure (CVE) and CCE misconfigurations), National Checklist Program (automatable security configuration guidance in XCCDF and OVAL), SCAP and SCAP-compatible tools, Product dictionary (CPE), Common vulnerability Scoring System for impact metrics, and Common Weakness Enumeration (CWE).

The Common Vulnerability Scoring System (CVSS) [26] is a vendor-neutral open source vulnerability scoring system. It was established to help organizations to efficiently plan their responses regarding security vulnerabilities. The CVSS is comprised of three metric groups classified as base, temporal, and environmental. The base metric group contains the quintessential characteristics of a vulnerability. The temporal metric group is used for non-constant characteristics of a vulnerability, and the environmental metric group defines the characteristics of a vulnerability that are tightly related to the user's environment.

6.3 VBAM and CVSS

We aim to make use of the CVSS to demonstrate the suitability of our proposal. We suppose that an OpenStack environment was installed in an Internet Engineering laboratory; we have to use VBAM to determine which group of users (faculty, Ph.D. students, master students) can access to which OpenStack services in the cloud environment. OpenStack is the most popular open-source cloud management platform. In this study, we consider ten of its services, which are briefly described hereafter. Dashboard, also called *Horizon (H)*, provides a web portal for the management of the underlying OpenStack services. Compute or *Nova (N)* facilitates the management of OpenStack's instances. Networking, codenamed *Neutron (Ne)*, this service, not only permits network connection between OpenStack's services, but also allows users to configure networks by putting an API into their disposition. Object Storage helps with the storage and retrieval of arbitrary unstructured data objects. It is also known as *Swift (S)*. Block storage or *Cinder (C)* provisions persistent block storage to running instances. The identity service is responsible of the identity management (authentication, authorization, endpoints) for the rest of OpenStack's services. This service is codenamed *Keystone (K)*. The image service, codenamed *Glance (G)*, takes charge of the storage and the retrieval of virtual machine disk images. Telemetry, codenamed *Ceilometer (Ce)*, helps monitoring and metering the business aspects of OpenStack like billing or benchmarking. Orchestration, or *Heat (H)*, facilitates the orchestration of multiple composite cloud applications. The Database as a Service, named *Trove (T)*, provides a scalable and reliable cloud database provisioning functionality.

Table 2 Vulnerabilities of the different services of OpenStack.

	H	G	Ne	N	K	S	C	T	Ce	H
AS	4.3	4.4	5.1	4.3	4.8	5.3	3.4	2.1	3.5	4.1

We know that the CVSS provides vulnerability scores that range between 0 to 10 and following the NVD severity ranking we have: LOW (0 – 3.9), MEDIUM (4 – 6.9), HIGH (7 – 10). We apply the same classification to the users of the laboratory to get the following:

- Master students → LOW
- Ph.D. students → MEDIUM
- Faculty members → HIGH

This means that users with a security level of LOW can only access objects with a similar security level. Users with a security level of MEDIUM, can access objects with security levels of MEDIUM and LOW. Finally, users with a security level of HIGH, can access all the services. We conducted a static method by retrieving from the NVD all the vulnerabilities of the different services of OpenStack that are in play in this paper, and their respective vulnerabilities. These details are recorded in **Table 2**. The last row of the table represents the average score (AS) of the vulnerabilities of the different services. Henceforth, a straightforward analysis of the table reveals that VBAM allows the faculty members to access all the services while the master students can only access *Cinder*, *Trove* and *Ceilometer*.

7. Discussion

In this section, we elaborate on different scenarios in which RAdAM could be used. Furthermore, we propose a solution to thwart some of the limitations of RAdAM.

In this research, our goal is to focus on the algorithm we propose for a risk aware authorization mechanism. Risk aware authorization mechanisms are required for modern systems that are highly dynamic; cloud computing is an example. We proposed different types of algorithms that, we deem, fit perfectly the entire ecosystem of cloud computing. We start from the principle of risk aware access decisions that every access request is dynamically evaluated. In the case of simple cloud scenario, where we consider that a user requests the basics services of a cloud service provider, we proposed a simple algorithm that encompasses all the intricacies of an access decision. We introduced parameters that are used in the algorithm for making the decision. To the best of our knowledge, we are the first to introduce the average parameters (average risk and average risk threshold) for situational factors. Additionally, many researchers define a risk formula for a given user, and the access decision uses the risk value regardless of the object that the user is trying to get access to and they also have the same threshold for all the objects that a particular user tries to get access to. In our model, every object has a risk that is associated to it. We contend that this is a more fine-grained type of authorization mechanism that can provide to a user different access levels to different objects. The average parameters we defined give an overall behavior of the user. Our algorithm is also versatile as we showed how it could be employed in collaboration and federation in cloud computing. We believe that the main weakness of RAdAM resides in the fact that we did not take into account many aspects of cloud federations, especially with regards to federated identity management. In the future, we will try to find how to integrate RAdAM in federation mode with federated identity management. The main issue with risk aware access control in general is the cost of computation. Our proposal does not solve the issue. The excitement about risk-aware access control revolves around the fact that in each access request, the risk will be evaluated. This certainly helps in making better access decisions but it is costly in terms of resource consumption. Furthermore, as stated by Ni et al. [15], an attacker can launch attacks in the time window between the access request and risk mitigation. In the future, we will try to see how we can leverage the computing power offered by cloud computing to resolve this issue. We believe that if our proposal is implemented in a system with 'unlimited' computer power, as cloud computing offers, the issue will become obsolete.

We also proposed VBAM, a vulnerability (or risk) based authorization mechanism that can cope with the dynamicity of modern computer systems like cloud computing. We contend that the future of access control revolves around tailoring the access decisions to the vulnerability score of the objects. We are aware that most attacks are the results of exploited vulnerabilities. Therefore, to limit the damages of a probable attack, we must adopt VBAM and other similar authorization mechanisms. One of the main issues of risk-aware access control in general is computational cost: our proposal is not an exception to that rule. Indeed, the fact that the system is at work every time a user makes a request results in high resource usage. The second point of contention relates to how our method can be used in an environment. We contend that it can be used in a kind of 2-step access control. Where VBAM will be used as primary authorization mechanism and a second authorization mechanism will decide whether the user can read write or execute the objects that he/she is allowed to get access to by VBAM. One might argue that our proposal does not solve the problem of authorization mechanism because a user with a high security level can exploit a vulnerability that has an equally high damage factor. While this is true, we argue that, in our system, users with a high security level are trusted users and they should be a limited number. Therefore detecting the culprit in any of these types of attacks is easy, as we already know the possible damages that each user can make. The same applies for the other security levels. As future work, we seek to implement our system in combination with the XACML framework in order to provide a more flexible authorization mechanism for the cloud.

8. Conclusion

We proposed a risk-aware authorization mechanism flexible enough to deal with the dynamicity of cloud computing. We proposed algorithms for simple user access, collaboration and federation modes. To complement RAdAM, we proposed a variation of MLS model where the risk is represented by the vulnerability score of the objects. We showed the applicability of our methods by using fuzzy inference systems for RAdAM and by proposing an OpenStack use case for VBAM. We contend that a combination of authorization mechanisms similar to the one we proposed in this paper represents the future of highly dynamic systems.

References

[1] Mell, P. and Grance, T.: The NIST definition of cloud computing (2011).

[2] Jason, P.O.: Horizontal integration: Broader access models for realising information dominance, Technical report, Report JSR-04-132, The MITRE Corporation, JASON Program Office, (2004), available from ⟨http://fas.org/irp/agency/dod/jason/classpol.pdf⟩ (2004).

[3] McGraw, R.: Risk adaptive access control (RAdAC), *Proc. NIST & NSA Privilege Management Workshop* (2009).

[4] Zadeh, L.A.: Fuzzy sets, *Information and control*, Vol.8, No.3, pp.338–353 (1965).

[5] Bell, D.E. and LaPadula, L.J.: Secure computer systems: Mathematical foundations, Technical report, DTIC Document (1973).

[6] Lindqvist, H.: Mandatory access control, *Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901*, Vol.87 (2006).

[7] Downs, D.D., Rub, J.R., Kung, K.C. and Jordan, C.S.: Issues in discretionary access control, *2012 IEEE Symposium on Security and Privacy*, pp.208–208, IEEE Computer Society (1985).

[8] Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E.: Role-based access control models, *Computer*, Vol.29, No.2, pp.38–47 (1996).

[9] Hu, V.C., Kuhn, D.R. and Ferraiolo, D.F.: Attribute-based access control, *Computer*, No.2, pp.85–88 (2015).

[10] Fall, D., Okuda, T., Kadobayashi, Y. and Yamaguchi, S.: Toward quantified risk-adaptive access control for multi-tenant cloud computing, pp.1–14 (2011) (online), available from ⟨https://sites.google.com/site/jwis2011/program⟩.

[11] Santos, D.R.d., Westphall, C.M. and Westphall, C.B.: Risk-based dynamic access control for a highly scalable cloud federation, *SECURWARE 2013, The Seventh International Conference on Emerging Security Information, Systems and Technologies*, pp.8–13 (2013).

[12] Santos, D.R.d., Westphall, C.M. and Westphall, C.B.: A dynamic risk-based access control architecture for cloud computing, *Network Operations and Management Symposium* (*NOMS*), *2014 IEEE*, pp.1–9, IEEE (2014).

[13] Rissanen, E.: Extensible access control markup language (xacml) version 3.0 (2013).

[14] Cheng, P.C., Rohatgi, P., Keser, C., Karger, P.A., Wagner, G.M. and Reninger, A.S.: Fuzzy multi-level security: An experiment on quantified risk-adaptive access control, *IEEE Symposium on Security and Privacy, 2007. SP'07*, pp.222–230, IEEE (2007).

[15] Ni, Q., Bertino, E. and Lobo, J.: Risk-based access control systems built on fuzzy inferences, *Proc. 5th ACM Symposium on Information, Computer and Communications Security*, pp.250–260, ACM (2010).

[16] Burnett, C., Chen, L., Edwards, P. and Norman, T.J.: TRAAC: Trust and risk aware access control, *2014 12th Annual International Conference on Privacy, Security and Trust* (*PST*), pp.371–378, IEEE (2014).

[17] Khambhammettu, H., Boulares, S., Adi, K. and Logrippo, L.: A framework for risk assessment in access control systems, *Computers & Security*, Vol.39, pp.86–103 (2013).

[18] Shaikh, R.A., Adi, K. and Logrippo, L.: Dynamic risk-based decision methods for access control systems, *Computers & Security*, Vol.31, No.4, pp.447–464 (2012).

[19] Hunter, J.S.: The exponentially weighted moving average, *J. Quality Technol.*, Vol.18, No.4, pp.203–210 (1986).

[20] Celesti, A., Tusa, F., Villari, M. and Puliafito, A.: How to enhance cloud architectures to enable cross-federation, *2010 IEEE 3rd Interna-*

tional Conference on Cloud Computing (*CLOUD*), pp.337–345, IEEE (2010).
[21] Buyya, R., Ranjan, R. and Calheiros, R.N.: Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services, *Algorithms and Architectures for Parallel Processing*, pp.13–31, Springer (2010).
[22] Goiri, I., Guitart, J. and Torres, J.: Characterizing cloud federation for enhancing providers' profit, *2010 IEEE 3rd International Conference on Cloud Computing* (*CLOUD*), pp.123–130, IEEE (2010).
[23] Recordon, D. and Reed, D.: OpenID 2.0: A platform for user-centric identity management, *Proc. 2nd ACM Workshop on Digital Identity Management*, pp.11–16, ACM (2006).
[24] Tang, B., Li, Q. and Sandhu, R.: A multi-tenant RBAC model for collaborative cloud services, *11th Annual International Conference on Privacy, Security and Trust* (*PST*), pp.229–238, IEEE (2013).
[25] Computer Security Division: National Vulnerability Database, US National Institute of Standards and Technology (online), available from ⟨http://www.nvd.nist.gov⟩ (accessed 2014-09-10).
[26] Mell, P., Scarfone, K. and Romanosky, S.: A complete guide to the common vulnerability scoring system version 2.0, *FIRST-Forum of Incident Response and Security Teams*, pp.1–23 (2007).

Doudou Fall received his M.E. degree in Data Transmission and Information Security from University Cheikh Anta Diop of Dakar, Senegal in 2009. He also received his M.E. and Ph.D. degrees in Information Science from Nara Institute of Science and Technology (NAIST), Japan in 2012 and 2015, respectively. He is currently an Assistant Professor in the Graduate School of Information Science, NAIST. His research interests include cloud computing security, vulnerability and security risk analysis.

Takeshi Okuda received his M.E. and D.E. degrees in Information Science from Osaka University, Japan in 1998 and 2011 respectively. He is currently an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. His research interests include virtual machine, virtual network and their security. He is a member of IEEE.

Youki Kadobayashi received his Ph.D. degree in Computer Science from Osaka University, Japan, in 1997. He is currently an Associate Professor in the Graduate School of Information Science, Nara Institute of Science and Technology, Japan. Since 2009, he has also been working as an associate Rapporteur of ITUT Q.417 for cybersecurity standardization. His research interests include cybersecurity, web security, and distributed systems.

Suguru Yamaguchi received his M.E. and D.E. degrees in Computer Science from Osaka University, Japan, in 1988 and 1991, respectively. Since 2000, he has been a Professor in the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). He has been working aggressively for making and running JPCERT/CC since 1996, and APCERT since 2002. From 2004 to 2010, he was appointed Advisor on Information Security to the Cabinet, government of Japan. His research interests include technologies for information sharing, multimedia communication over broadband channels, large-scale distributed computing systems including cloud computing technology, network security and network management for the Internet.