A $\mathrm{UCON}_{\mathrm{ABC}}$ Resilient Authorization Evaluation for Cloud Computing

Arlindo Luis Marcon Jr., Altair Olivo Santin, *Member*, *IEEE*, Maicon Stihler, and Juliana Bachtold Jr.

Abstract—The business-driven access control used in cloud computing is not well suited for tracking fine-grained user service consumption. $\mathrm{UCON}_{\mathrm{ABC}}$ applies continuous authorization reevaluation, which requires usage accounting that enables fine-grained access control for cloud computing. However, it was not designed to work in distributed and dynamic authorization environments like those present in cloud computing. During a continuous (periodical) reevaluation, an authorization exception condition, disparity among usage accounting and authorization attributes may occur. This proposal aims to provide resilience to the $\mathrm{UCON}_{\mathrm{ABC}}$ continuous authorization reevaluation, by dealing with individual exception conditions while maintaining a suitable access control in the cloud environment. The experiments made with a proof-of-concept prototype show a set of measurements for an application scenario (e-commerce) and allows for the identification of exception conditions in the authorization reevaluation.

Index Terms—Access controls, security and privacy protection, distributed systems

1 Introduction

Employing the traditional approach for policies definition, configuring statically authorization attributes in the policies rules for each user, may lead to a scenario in which a given user's resource would be underutilized, while for another user it would be in shortage. There is no way for precisely anticipating how much consumption demand a given user will generate on a service, in such way that matches the policy and each user attributes' requirements. Thus, the evaluation model should be suitably resilient to accommodate particular policies exceeding without impairing other users or the whole access control system [1], [2].

A real-word example that shows this fact is an e-commerce web service site hosted on a cloud computing platform. It can experience sudden spikes on the number of incoming requests due to some promotional campaign. In the business-driven access control, the virtual machine (VM) instances responsible for the e-commerce web service are usually initialized with the usage policies stored locally and statically defined, but the web service will be used by many different users from different domains.

Each user will interact with the e-commerce web service, demanding different amounts of computational resources. However, in the operating system level, usage accounting for computing resources will be made for the web service as a single account. As the web service's users are only known at the service entry point (software as a service, SaaS), it is not possible to measure the service consumption corresponding to each user. An additional disadvantage of business-driven access control is the impossibility of offering different user experiences, because the user identity cannot be easily known by the different services [3]. Thus, coarse-grained usage control can favor the misuse by some user.

A reason to deploy a security system based on fine-grained access control is to facilitate the usage management tasks. For cloud computing, it should be scalable, effective, and flexible for providers and consumers [4].

A fine-grained user service accounting and authorization attribute reevaluation approach allows the consumer to control individual usage of services hosted on the providers [5].

A consumer may contract the same service from different cloud providers. In this context, it is necessary to standardize the access to cloud services. However, there is no consensus on standardization; thus, web services can be one alternative to facilitate the interaction in heterogeneous and scalable environments [6].

Providers are expected to honor service-level agreements (SLA) established with an "intermediary contractor" (i.e., broker) [7] and to control access to the consumer's and system's data.

In cloud computing, it is desirable to do the accounting for services usage frequently, to reevaluate authorization and to detect usage disparity, dynamically reconfiguring the policies. This assures an even utilization of the computing resources, avoiding the prejudice to user experience [8]. However, the task of rewriting policies periodically may impose an important burden on the usage control system. Additionally, the user consumption can span over many virtual machines (service instance). Therefore, balancing a given user's policies (authorization requirements) among specific VMs becomes a challenge during reconfiguration.

• *The authors are with the Graduate Program in Computer Science, Pontifical Catholic University of Parana, Curitiba 81530-150, Parana, Brazil. E-mail: {almjr, santin, stihler, jbachtold}@ppgia.pucpr.br.*

Manuscript received 21 Sept. 2012; revised 19 Mar. 2013; accepted 29 Mar. 2013; published online 9 Apr. 2013.
Recommended for acceptance by X. Li, P. McDaniel, R. Poovendran, and G. Wang.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2012-09-0976.
Digital Object Identifier no. 10.1109/TPDS.2013.113.

The $\mathrm{UCON}_{\mathrm{ABC}}$ usage control enhances the classical access control by reevaluating continuously the user attributes during consumption of a service or resource against the usage policies [9]. The usage can be understood as an object's (e.g., a file) read and write operations and resource consumption (e.g., CPU cycles) [10].

The periodicity (frequency) a given user is accounted interferes directly on the continuity of the policy evaluation process on $\mathrm{UCON}_{\mathrm{ABC}}$. Thus, the periodicity of reevaluation defines the maximum amount of time a given user may be in a condition that misleads the usage control, characterizing an exception (usage authorization disparity).

Obviously, it is desirable to obtain the smallest interval of time for obtaining usage attributes and reevaluating the usage policies. However, when frequency is very high, the service will suffer a significant overhead. There are no proposals dealing with these issues in the literature.

Finding out the best accounting reevaluation frequency for the resources is very important. The best frequency means that it will not cause service overhead while reducing the period of possible authorization inconsistency. The environment for managing usage must be flexible enough to adapt itself to the different needs of a consumer domain (CD). The usage data collected from the involved providers should be used to feed the usage attribute repository and the reference monitor that reevaluates usage control policies for each user.

For each consumer's service purchased, an SLA is required to make sure that the contracted resources amount will be correctly provided [11]. Therefore, the access control management, in a cloud-based architecture, should support a monitor collecting accounting usage attributes—the accounting agent—on an arbitrary number of providers, in a scalable fashion.

It is important to notice that a fine-grained usage control is not tackled by the cloud computing, given it is considered a consumer responsibility and is not addressed on any literature proposal. However, if a consumer purchases an infrastructure service to concentrate its effort in the development of its business, it is not desirable for this consumer to develop a customized access control architecture herself. Therefore, it would be better to have an access control model provided as a platform as a service (PaaS) to deal with the security issues.

The policy evaluation system and the attribute repositories, on this proposal, employ a distributed shared memory implemented on a tuple space service [12], hosted on a pool of servers. In this manner, the environment for accounting and policy reevaluation can be scaled to the usage demands, tweaking the architecture to the requirements of each consumer and set of cloud providers. The flexibility of the evaluation system and attribute repositories is transparent to consumer and provider entities, impacting positively on the expansion or reduction of the access control system. This usage control feature is not present on any literature proposal.

The proposed approach provides a resilient $\mathrm{UCON}_{\mathrm{ABC}}$ reevaluation authorization model for cloud computing. The usage architecture for collecting contextual data allows for fine-grained services accounting and authorization attributes. The data are consolidated on a management domain and provided to consumer management systems, enabling the reconfiguration of usage policies and monitoring of SLA fulfillment. The management services are provided through a federated environment (FE) hosted on a cloud computing provider. The federation environment is a usage control management domain shared by the policies evaluation system, accounting attributes handling system, SLA manager and service's broker. The service's broker offers an entry point for cloud users and an FE entry point for providers and consumers.

This work is organized as follows: Section 2 briefly introduces cloud computing and access control for dynamic environments; Section 3 shows the related works; in Section 4, the proposal's model is presented; Section 5 addresses the proposal's architecture; Section 6 presents the prototype and evaluation tests; Section 7 draws the proposal's conclusions.

2 Cloud Computing and Access Control

In this section, some aspects regarding cloud computing are discussed briefly, as well as the access control applied on dynamic and heterogeneous environments.

2.1 Cloud Computing

The client-server architecture and commercial off-the-shelf software are still dominant on today's computing landscape. However, cloud computing is changing this paradigm, offering services and computational resources as utilities [13]. On cloud computing, the consumer (services buyer) requests resources on demand as the need arises, without being concerned about infrastructure management. This approach affects the computational ecosystem as a whole (e.g., from the infrastructure to the users) [14].

A cloud service hosted on computational centers should be available and accessible through the network (e.g., Internet). The consumer does not need to be bothered by infrastructure management complexities as in client-server architecture.

The main entities involved in cloud computing can be briefly described as: 1) resource or service providers (SPs)—offer computational services (e.g., infrastructure, platform, and software); 2) consumer—entity that purchases cloud services (usually to provide it for her users); 3) user—entity who consumes the service or resources, the end user.

Cloud providers can be categorized according to the service model [15] as (i) *Software as a Service* (SaaS): It is the final cloud computing product—on this service, the consumer does not control the underlying system, although it is possible to use the applications hosted on it; (ii) *Platform as a Service* (PaaS): offers resources for consumers to deploy their own applications—the consumer does not manage the underlying infrastructure (e.g., network, storage); however, she has control over her applications and the system that controls the environment configurations; and (iii) *Infrastructure as a Service* (IaaS): provides basic computational resources (e.g., processing power, storage space) for consumers to deploy their own operating systems—on this service, the consumer does not handle the physical

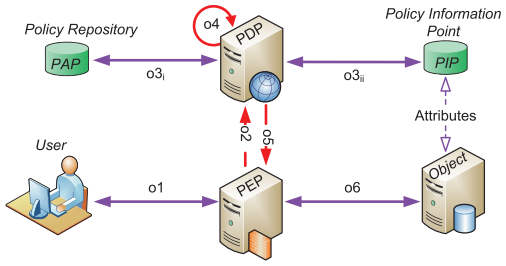

Fig. 1. Access control architecture for heterogeneous environments.

infrastructure, but she has partial control over network components (e.g., *host firewalls*).

The IaaS employs virtualization technologies to make flexible the use of the underlying hardware [16]. The VM is the mechanism that provides an independent computational environment. A VM may be instantiated and destroyed under demand, while the cloud computational services are being provided [17].

2.2 Access Controls for Dynamic Environments

The most employed approach to evaluate access control policies, on heterogeneous and distributed environments, is based on the reference monitor and guardian (active entities in the authorization evaluation system) and the policy repository (passive one) [18]. The reference monitor (policy decision point—PDP) evaluates the policies it retrieves from the policy repository (Policy Administration Point—PAP) against the subject's (user) attributes to decide if an access to an object should be allowed or not. The guardian (policy enforcement point—PEP) executes a PDP decision.

In the outsourcing model (Fig. 1), the PEP receives an access request (*event o1*) from a user and forwards it to the PDP (*event o2*). The PDP retrieves policies from the PAP (*event* $o3_i$) and attributes from the policy information point (PIP; *event* $o3_{ii}$), and decides (*event o4*) if the requested access should be allowed or denied, thus communicating such decision to the PEP (*event o5*). The PEP executes actions to enforce the PDP decision (*event o6*).

The access control model that best suits the needs of dynamic environments is the usage control UCON_{ABC}. The UCON_{ABC} usage controls reevaluate authorizations periodically, taking into account the attributes mutability (ongoing updates in the usage accounting and authorization attributes). Authorizations may be understood in the traditional form of granting and evaluation of rights (e.g., read, write).

The UCON_{ABC} evaluates its controls continuously, because it understands that attributes of user and object (resources or services) may change (mutability) during the time an object is under use. An example of mutable attribute is the user quota in a file system or storage.

Mutable attributes may be updated before or during an object usage. The constraint model considers two stages for controls evaluation: before (pre) and during (ongoing) a resource or service usage. Evaluation before usage characterizes the classical authorization procedure, while evaluation during ongoing usage is required by the attribute mutability scheme.

3 Related Works

The proposal of Zhang and Zhang [19] aims at helping SPs and consumers to increase flexibility, extensibility, and adaptability of the services being provided or used on the cloud. That approach offers an access interface that is homogenous for the consumer, and it may be composed by different SPs. However, the proposal adopts a traditional security scheme that does not follow the flexibility offered by the SaaS service level.

Lim et al. [20] consider that control policies together with data collected by the provider may help the consumer to better allocate resources for usage. To enable the usage of dynamic controllers outside the cloud, it must expose sensors and actuators, allowing the enforcement of policies. However, sensors and actuators, with help from an intermediating domain responsible for collecting and consolidating fine-grained data from the environment, may provide a more adequate solution to managing the environment.

Bertram et al. [21] argue that a service-oriented infrastructure, at the PaaS, is one solution for security risk management related to the assets (e.g., goods or services) shared on collaborative clouds. Despite proposing the employment of attributes dynamically generated by the environment, executing the creation and automated mapping of those attributes to control policies, it is not clear how this process is accomplished.

The PESA architecture, proposed by Goyal and Mikkilineni [22], allows a management approach based on policies to control the characteristics of a service (e.g., availability, performance, security, risk management). PESA contemplates mainly the management of resources being used on the cloud. However, the evaluation of user access requests or the control over who is making use of a service is not covered by the architecture.

Tavizi et al. [23] proposed an architecture to use UCON_{ABC} in cloud computing. The paper presented a scheme to address obligations and eXtensible Access Control Markup Language (XACML) extensions to support the need of UCON_{ABC} in cloud computing. The authors described the main entities of the proposal, but did not consider implementation and evaluation issues.

In the proposal of Danwei et al. [24], it is described a negotiation module that is activated when the user has no suitable attributes to access a service. The proposal is based on security assertion markup language for security assertions and XACML for policies language. The paper is theoretical, and the presented approach is applied only to local environments.

Services hosted on the cloud are distributed and dynamically provided. Traditional approaches to address access control, as those described previously, are not adequate for these environments. Cloud computing requires an approach for evaluation of usage policies that allows a distributed service accounting, in a fine-grained manner, and taking into account periodical evaluation of policies to maintain the whole system's access control integrity. This policy evaluation environment and consumption accounting service must be flexible enough to deal with the cloud's elasticity.

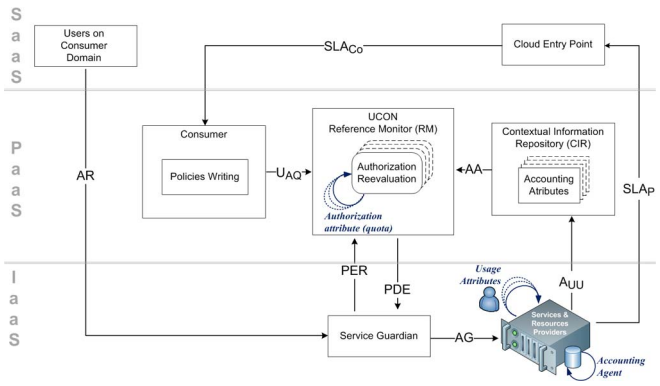

Fig. 2. Proposed $\mathrm{UCON}_{\mathrm{ABC}}$ ongoing authorization model.

4 Proposed Model

The proposed model aims to extend the $\mathrm{UCON}_{\mathrm{ABC}}$ ongoing authorization model, providing resilience to the reevaluation of usage policies. Resilience means providing the model with the ability to deal with some individual user authorization attributes exceeding, while the SLA (Fig. 2, *event* SLA_{Co}) for the respective consumption service is under the contracted amount. The consumer writes policies (*event* U_{AQ}) for her users, defining individual authorization attributes—a user's service usage quota (U_{AQ}). Quota is used during the reevaluations in place of authorization attributes; the aim is to relax the usage policy when possible.

The contextual information is obtained for each user's service usage (A_{UU}), i.e., the user accounting attributes are obtained from providers through the accounting agent. Therefore, the resilience for ongoing authorization reevaluation (R_{AR}, (1)) is defined only if SLA_{Co} minus the sum of each user attribute accounting (*sua*) is greater than t. The constant t is a spare quota, freely defined by the consumer for that service or resource

$$R_{AR} \exists \left[\left(SLA_{CO} - \left(\sum_{i=1}^{n} A_{UU_i} \right) \right) > t \right]. \quad (1)$$

That means, when the users' consumption (*sua*) is close enough to the contracted amount ($SLA_{Co} - t$), a new SLA with additional quota should be negotiated to avoid an SLA violation.

Next, for the sake of simplicity, it is shown a simple scenario, involving a file system, aiming to show how quota is used in the model to provide resilience to $\mathrm{UCON}_{\mathrm{ABC}}$ authorization.

Considering a consumer that negotiates an SLA_{Co} for a service, for example, storage space: 600 GB and t: 100 GB. The consumer writes usage policies for userA: 200 GB, userB: 200 GB, and userC: 100 GB. When a user requests an access to the provider (Fig. 2, *event AR*), the service guardian sends an authorization evaluation request for such user to the Reference Monitor (RM; *event PER*). In its turn, the RM sets up the usage quota for that user (initially equal to the authorization attributes, e.g., userA: 200 GB, userB: 200 GB, and userC: 100 GB) and provides an "allow" response to the access control enforcement (*event PDE*).

After getting the provider's access released by the enforcement (service guardian, *event AG*), the user starts the storage consumption. Therefore, the accounting agents send the usage attributes of each user (e.g., userA: 190 GB, userB: 10 GB, and userC: 0 GB) to the Contextual Information Repository (CIR). After a while, the service guardian requests an authorization reevaluation. In the reevaluation (ongoing authorization), the RM compares each user's accounting attribute (*event AA*) against the quota (mechanism to implement authorization resilience). In such a case, the storage consumption of each user is below their respective quotas.

A period of time after, the accounting agents send to CIR the consumption of each user again (e.g., userA: 250 GB, userB: 20 GB, and userC: 10 GB) and the reevaluation is required. During the RM's authorization reevaluation, it is detected that userA's accounting attribute is exceeding userA's quota; thus, computing *sua* yields 280 GB. As the $SLA_{Co} - t$ allows 500 GB of storage consumption for the consumer and userA is exceeding the attribute authorization threshold, the quota is automatically expanded as userA: 250 GB.

After a while, the accounting agents send the usage attributes of each user again (e.g., userA: 240 GB, userB: 160 GB, and userC: 100 GB) to CIR. During the RM's authorization reevaluation, requested by the guardian, it is noticed that userA's usage attribute is below userA's quota, but computing *sua* yields 500 GB of storage consumption. As the R_{AR} is not valid ($SLA_{Co} - sua = t$) and userA quota is exceeding the usage policy, it should be made equal to the authorization attribute (usage policy) again, userA: 200 GB. If in the next reevaluation the usage attributes reported by accounting agents keep a value over the quota and *sua* is close enough to SLA_{Co}, userA will be under an exception condition. Otherwise, userA's quota can be tweaked again.

The exception conditions occur when a user, during the reevaluation, is detected exceeding a quota that was authorizing her in the previous evaluation. The exception conditions happen because during the reevaluation period (current and previous) the user continues to consume services. Moreover, such situation can be triggered by the resilience of the model, which resets the quota to the original authorization attribute value. An exception may also occur because during the reevaluation period there was a change that made a policy more restrictive than the one previously defined.

When a user is detected in an exception condition, the consumer must rewrite the usage policies to bring the user's condition back to normal, as before the exception was reached. The exception condition may lead the consumer to conclude, when the *sua* is close to SLA_{Co}, that more service needs to be contracted (if available in the provider, Fig. 2, event SLA_P). Also, the elasticity provided by the cloud can be used in an automatic request for more service—when it is depleting, thus allowing the model's resilience to be continuously provided.

One can notice that in traditional cloud approaches, when a user reaches the quota, there is no support for exceeding authorizations in the model. In fact, the consumer's manager needs to look for a user that is not using

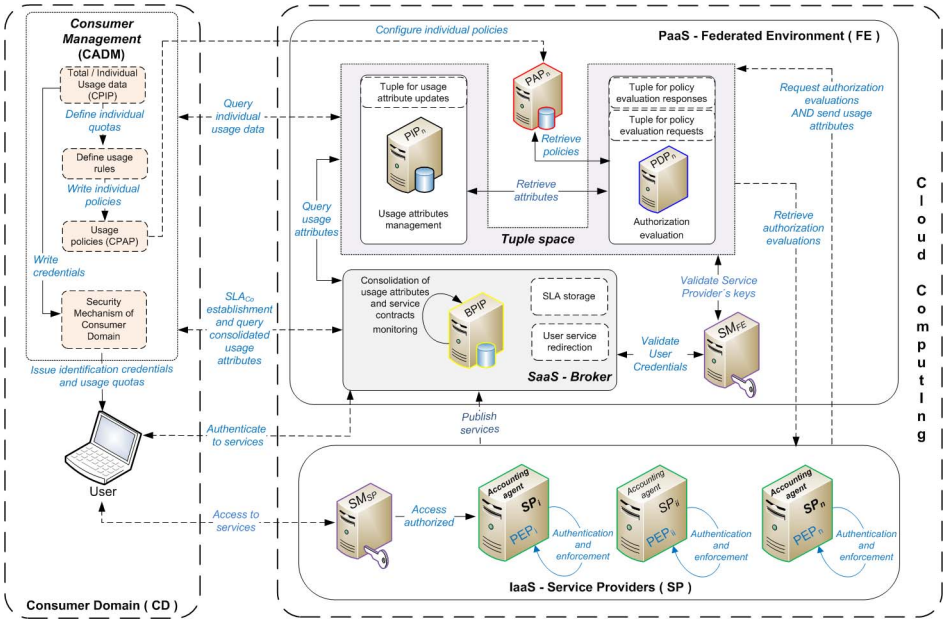

Fig. 3. Proposed architecture overview.

its allowed quota and rewrite the policies. However, it is a manual and tiresome task performed by a human without having real consumption attributes to help. Therefore, there is no support to decide how to change the policy for a target user. Applying the proposed resilient model provides an automatic authorization threshold balance, and only after computing *sua* and identifying that SLA_{Co} has been reached, the consumer's manager is required to act. Therefore, the reevaluation provides authorization resilience, exploring idleness on the service consumption, if the user's *sua* is below the threshold ($SLA_{Co} - t$).

In traditional approaches, authorization is only evaluated in the beginning of a usage (preauthorization); thus, there may be inconsistencies between the authorization granted and the active authorization attribute. Notice that even the ongoing reevaluation of authorization is not enough to make sure that some authorized usage for a given user will not be violating a policy. This may happen because exception conditions may occur between reevaluations. However, in this proposal, those periods will be smaller than in traditional approaches, depending only on the time interval between reevaluations.

In Fig. 2, the resilience can be seen as dashed lines in usage attribute and authorization attribute, as presented before. Furthermore, the elasticity provided by cloud computing is used to automatically add reference monitors and CIR instances when the demand for authorization reevaluation and accounting attribute retrieval increases.

One can notice that the proposal as a cloud PaaS frees the consumer of the responsibility of dealing with security aspects; therefore, she can focus on the development of services for its own business.

5 Proposed Architecture

The proposed architecture is based on a computing cloud composed of several types of services, providers, and consumers, and each of these may be exchanging data between different services (SaaS, PaaS, and IaaS). An intermediating (federated) environment is used, aiming to ease the interaction between entities using the cloud computing and to offer a PaaS that is $\mathrm{UCON}_{\mathrm{ABC}}$ security oriented with ongoing authorizations (Fig. 3).

An entry point to the cloud computing environment for providers, consumers, and users is available through the broker (Fig. 3). Broker intermediates the offering of services by providers, the consumer's negotiations for obtaining service access as well as for defining SLAs, and redirects consumer's users to the network address of SPs.

The SP advertises itself as a provider for the federated cloud environment through the broker. For each offered service, the provider negotiates an SLA with the broker and sends to it a service interface description. An SLA defines the service agreed items, which will be a source for the broker to derivate the service amounts offered to consumers. A service interface description will be provided to the consumer's domain users, allowing them to implement their systems, compatible with the services offered by the SPs.

The consumer establishes an SLA with the broker, who represents the federated providers, and defines usage policies for the users of her own domain, regarding the contracted services. The usage control policies will be configured on a PAP on the FE (Fig. 3).

The established SLA provides each consumer with a set of services employed to manage the environment (e.g., services for storing attributes, policies, credentials, and cryptographic keys). Authentication credentials are created considering the consumer accounting, allowing the users to interact with the broker, so they can reach the desired SP.

The CD's user receives a quota defining the usage attributes ("rights") for service consumption. Before a user can exercise her quota, she must present herself to the broker. In the initial contact, the user must provide the broker with authentication credentials—an identification credential signed by the consumer. The broker selects the

provider that fits best the services or resources required. A reference (interface describing how to access the service) is returned as a response to the user, who will start to directly access the provider in future interactions. This approach allows the broker to redirect users to a specific provider, according to the SLA for each consumer.

The service requests received by the providers, which make up the cloud federation, are interpreted by the guardian (PEP) of each service and are evaluated by implementations of reference monitors (PDP) instantiated on the FE. The user's service accounting is stored by the CIR (PIP). The accounting attribute repository is a service updated by the accounting agents, running on the service providers of the federation (Fig. 3).

After the consumer configures usage policies, based on the SLA, and user starts the service consumption, the PIP begins to store accounting attributes for each user. Periodically, the broker consolidates usage attributes for each consumer to evaluate if the services' SLA_{Co} is being honored. On the CD, the consolidated usage attributes are analyzed to decide if a user's usage policy needs to be tweaked. Using this approach, the consumer is able to redefine usage policies according to the needs of each user, optimizing the use of service and minimizing the authorization exceptions.

The federation provides interoperability, high availability and elasticity, employing an approach based on distributed shared memory. The shared memory, implemented as a tuple space service, can be accessed by many accounting agents from the SPs to write user's usage attributes. Moreover, the tuple space is used for authorization reevaluation requests (from PEPs) and responses (from PDPs).

The policy evaluation environment is based on the outsourcing model. In the proposed approach, outsourcing means the evaluation is made on the FE. However, it is not in a centralized way, given it is applied a tuple space service supporting many PDPs working in parallel. According to our evaluation, the outsourcing model is more suitable to the context of cloud computing due to the dynamicity inherent to the environment (i.e., frequent instantiation and destruction of VMs). In this model, no time is spent transferring policies and managing cache systems on the provider's environment, for each instantiated service.

One can notice that for each service, the policies for the user from the consumer's domain should be configured. In many instantiated services, the "ephemeral requests" for the user policy evaluation does not justify the cost of transferring and managing the policy on the provider. In this proposal, the outsourcing approach stores the policies on the PAP, thus providing a better control over the policies, also avoiding policy disparity (inconsistences caused by cache of policies in provider's domain).

The expansion or retraction of policy evaluation environment occurs according to the environment's demand, i.e., according to the amount of consumers making use of the tuple space. Employing tuple spaces for sharing data on distributed systems, like cloud computing, aims to create a temporal decoupling between data consumers and providers. In this context, the entities that read and write the contents (tuples) can be asynchronous and does not need to know each other. The interaction can be carried away as long as the entities agree on the template form used to store data on the common space.

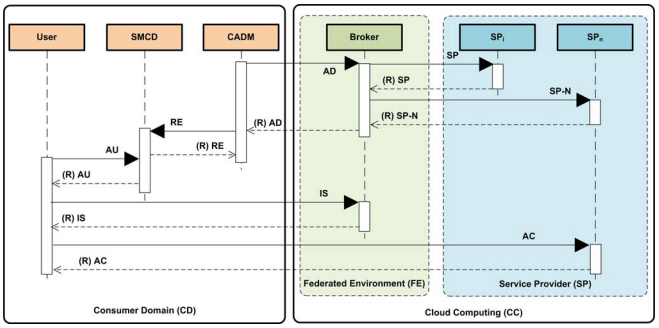

Fig. 4. The consumer in the proposed architecture.

The expansion and retraction of services happens automatically, and it is transparent to the consumer and provider. The more the consumers, the bigger will be the pool of servers for the tuple space service, because each consumer demands providers' services and management services for her own domain. The proposal's cloud-based approach provides high availability and elasticity, with low functional coupling between the environment's entities. This result is obtained using the cloud's infrastructure on the PDP, PIP, PEP, and the tuple space server pool.

5.1 Consumer Domain

The CD congregates the consumer management, an entity that contracts the offered services on the FE and the consumer's users for the contracted service.

The consumer's management (CADM) writes authentication credentials for its users on the security mechanism of consumer domain (SMCD, Fig. 4, *event RE*).

The definition of usage policies for the users of a CD is made based on a contracted SLA_{Co} obtained from the broker (Fig. 4, *event AD*). The policies define each user's individual usage constraints for using the services, and the access credentials the users will employ (*event AU*) to request the service reference on the broker (*event IS*). The reference informs the user the way to access and use the services on the providers (SP_n), which are associated with the federation (*event AC*).

5.2 Federated Environment and Providers

The FE, represented by the broker, offers to a user a single entry point to reach resources and services provided by all providers associated with a federation. The broker is the entity responsible for administrating the contracted SLA established with the consumer. When a user demands a service, the broker chooses among the federated providers the one that fits best the usage policies. It is the broker's duty to find out providers to satisfy the needs of consumers, it should apply scheduling policies for resource allocation to help in the choice, as proposed by Amit et al. [25].

The SP is an entity associated with the federation. The association process is managed externally to the context of this proposal. However, once associated, the provider starts to enjoy the service management and providing

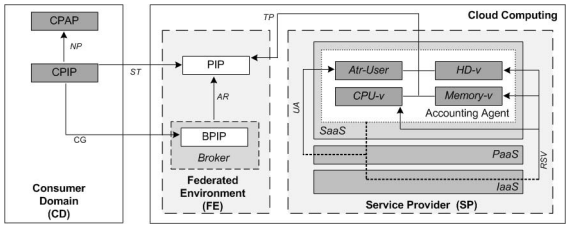

Fig. 5. Architecture for attributes management.

infrastructure of the FE. Resources and services on this environment may be scheduled according to the cloud's geographic scope or the type of service offered by each provider.

On the PaaS, the controlling services intermediates the web services access requests (SaaS), applying policies for each request from users belonging to the CD. When the access is granted on the PaaS, the user is allowed to use the resources being provided (IaaS). However, the outsourcing model adopted in the proposal frees the provider and the consumer from the task of implementing the $\mathrm{UCON_{ABC}}$ reference monitor and consumption attributes management, which is placed in the FE; policy enforcement (PEP) and accounting agents can be embedded in the consumer developed software.

5.3 Attributes

The attributes belonging to the IaaS refers to the total resources used by a VM (Fig. 5, *event RSV*) or service instance. These attributes are related to the resources allocated to the consumer (e.g., CPU-v, Memory-v, HD-v). The PaaS provides accounting attributes related to the users who are using the services (e.g., Attr-User, *event UA*).

The consumption attributes handled on the PIP are provided by the accounting agents running on the VMs on the SPs (*event TP*). The PIP provides information on the individual user's usage to the consumer PIP (CPIP, Fig. 5, *event ST*). The usage attributes stored on the PIP are also provided to the attributes manager on the broker (BPIP *event AR*). The broker consolidates the usage attributes for the whole consumer's domain, aiming at monitoring the SLAs.

Considering each user's attribute (Fig. 5, *event ST*) and consolidated usage (*event CG*), it is possible to identify which user is using the services on the provider (*event UA*), as well as to identify whether there is any idling resources on the provider's environment (*event RSV*). The usage attributes are provided to the consumer PAP (CPAP) (*event NP*); thus, it can generate new policies for CD users, justify the need for contracting more resources, or optimize the usage ratio of resources allocated on the provider. Therefore, from the consolidated usage attributes, it automatically obtains a balance among the specific instances of a service during the policies reconfiguration.

5.4 Security Mechanisms

To access the SP, each user receives a credential signed by the consumer (SMCD, Fig. 4, *event AU*). The credential allows the user to get the interface description to access the service. Moreover, it provides information about the access authorization needed to use the service.

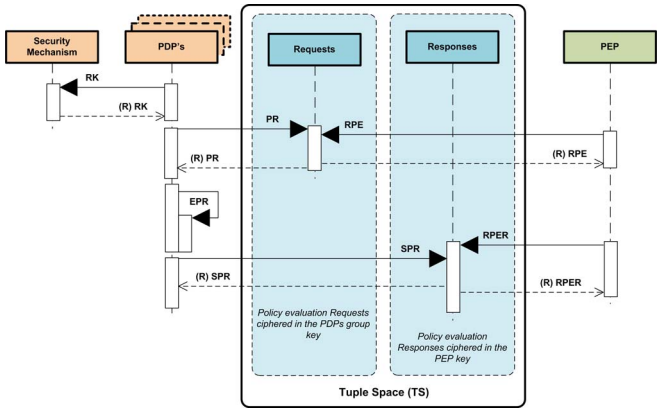

Fig. 6. Security in the tuple space-based policy evaluation.

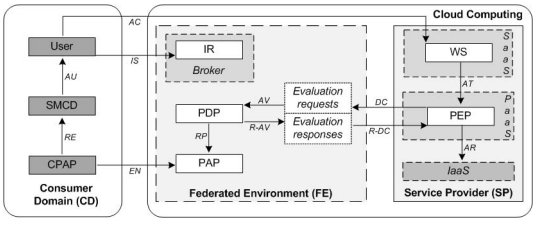

Fig. 7. Policies management system.

As the proposal is deployed in a distributed system, the policy evaluation system applies cryptography in the entities' interaction. The policy evaluation messages sent by the guardian (PEP, *event RPE*, Fig. 6) to the tuple space service are signed and encrypted with a group key shared by the PDPs, as proposed by Harney and Muckenhirn [26]. Any PDP of the group can process a new PEP reevaluation request that reaches the tuple space (*events PR, EPR*). The correspondent policy evaluation decision is encrypted with the PEP's public key and sent back to the tuple space service (*event SPR*, Fig. 6), after the PDP signed it with the group key. Every new PDP that integrates the cloud-based elastic authorization reevaluation system must have the group key to open the requests sent to the tuple space by the PEPs (*event RK*). The key's management and group membership is responsibility of the broker.

The accounting attributes sent by the accounting agent are signed and stored on the tuple space service, after being encrypted with the PIP's public keys. The procedures for signing and encrypting are responsibility of the credential's service on the provider (Fig. 3; $\mathrm{SM_{SP}}$).

Consumer and provider have a trust relationship with the broker. On the provider's environment, the user needs to be authenticated by the public key infrastructure of the credentials' service (Fig. 3; $\mathrm{SM_{SP}}$). The credential provided by the user (issued by the consumer) proves that she is part of the federation, as consumer and provider share a trust relationship with the broker.

5.5 Usage Control and Policies Management

The CPAP and SMCD (Fig. 7) convert SLA into rules for defining policies and access credentials (*event RE*) for users on the CD (*event AU*). Users from the CD access the interface repository (*event IS*) on the broker, and the respective services on the provider (WS, *event AC*) using those credentials.

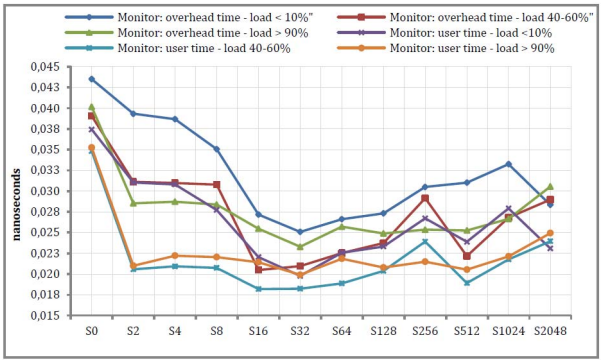

Fig. 8. The best periodicity of usage attributes monitoring.

TABLE 1
Stress Testing of Tuple Space

Tuple size (including 1KB of headers)	No. of tuples stored before crashing	Time spent to store a tuple (ms)	**Throughput** (KB/ms)
2 KB	69864	2,54	55011
4 KB	45333	2,57	70557
8 KB	26651	3,01	70833
16 KB	14605	3,40	68729
32 KB	7670	4,94	49684
64 KB	3935	7,90	31878
128 KB	1492	13,44	14209
256 KB	1002	25,39	10102
512 KB	502	48,84	5262
1024 KB	250	98,88	2589
2048 KB	124	200,83	1264

The policies, derived from an SLA, are stored on the CPAP and configured on the PAP (*event EN*, Fig. 7). The CPAP is a service used for managing access control policies internal to the CD, after those policies are transferred to the FE. The PAP receives and stores the policies sent by the CPAP, to be used afterward by the PDP (*event RP*) in authorization evaluation requests—performed by a PEP (*event DC*) running in a VM on a (federated) SP. Policies created by the consumer and stored in the PAP also are used to setup the user quota.

As the access control is implemented using an outsourcing model, the user must be authorized by the PDP and released on the PEP (*event AT*) to carry on the service usage (*event AC*, Fig. 7). The authorization evaluation process is executed on the federation (*event AV*). The policies retrieved from the PAP (*event RP*) are evaluated on the PDP. When the evaluation response is positive (access allowed, *event R-DC*, Fig. 7), the user is authorized to access the resource available on the IaaS (*event AR*), and this decision is enforced by the PEP.

The service's access requests received by the PEP (*event AT*) are forwarded to the tuple space service on the federation. The tuple space is constantly monitored by the PDPs. Thus, a PDP server (any reference monitor from the pool of PDP servers) retrieves the request from the tuple space service. After evaluation, it addresses the response to the requesting PEP through the tuple space service (event *R-AV*). The requesting PEP monitors the tuple space service to obtain the policy evaluation response. The authorization evaluation uses usage attributes available on the attribute repository (PIP, Fig. 5).

6 Prototype and Evaluation Test

The following section presents the technology used to implement a proof-of-concept prototype and to make experimental evaluations with it.

6.1 Prototype

The proposed prototype was implemented using a cloud operating system, web services, and tuple spaces. The cloud operating system employed on the proposed environment was the VMWare vSphere 4.1 (*vmware.com/products/vsphere/overview.html*). In the prototype, the attributes were accessed with help from the following application libraries: VMware Infrastructure Java API (*vijava.sourceforge.net*); APIs for accessing the Java virtual machine (JVM) provided by the Java development kit (*java.lang.management*); the projects JavaSysMon (*jezhumble.github.com/javasysmon*), and SIGAR (*hyperic.com/support/docs/sigar*). Usage policies were created and manipulated using XACML [18] from the Sun XACML API (*sunxacml. sourceforge.net*).

The services provided by the VMs are hosted on an Apache Tomcat (*tomcat.apache.org*), being accessed through the SOAP [27] engine Apache Axis2 (*axis.apache.org/axis2/java/core*). The Rampart module (*axis.apache.org/axis2/java/rampart*) integrated into Axis2 provides the security (signature and encryption) needed on the SOAP messages (specifications WS-Security [28] and WS-Trust [29]). The tuple spaces used to implement the FE's shared memory employ the technology provided by JavaSpaces, developed on the project River (*river.apache.org*).

The tests were performed on an environment that implements the proposal (Fig. 3) with the following specifications: The CD runs on an Intel Core i7 2-GHz CPU, 6 GB of RAM, and Windows 7 x64 as operating system. The provider's server employed an Intel Xeon 2.6-GHz dual processor with six cores, 48 GB of RAM, and 3 TB of storage. The FE server used an Intel 2.0-GHz dual processor with four cores, 16 GB of RAM, and 3.3 TB of storage. The servers were handled using VMware vSphere ESX 4.1, hosting the VMs and services described in this proposal (Fig. 3) for the two environments. The machines were connected through a Gigabit Ethernet network.

6.2 Evaluation Tests

The service offered to the user in this scenario is simulated as an e-commerce, running in a JVM, and the CPU load had to be synthesized using cryptographic algorithms. The key sizes were adjusted to create different CPU load ranges (load $< 10\%$, $40\% <$ load $< 60\%$ and load $> 90\%$). In this scenario, each accounting agent (Fig. 8) submits 17 different usage attributes to the tuple space.

The measurements (Fig. 8) were obtained running 1,000 iterations and computing the mean of them for each item presented in the graph; similar procedure was adopted in the case of Tables 1 and 2; the variance was below 5 percent in all cases. The tests were performed on a controlled local network environment; thus, it was possible to know the system's behavior without external interferences.

The CPU time was obtained using the class Java *ThreadMXBean*. The CPU load on the VM was measured using the SIGAR APIs and Java SysMon (Section 6.1).

TABLE 2
Stress Testing for PDP Policies Reevaluation

Policy size(retrieved from PAP)	No. of parallel policies reevaluated before crashing	Time spent to evaluate a policy (ms)
4 KB	1690	0,129916
8 KB	1360	0,129123
16 KB	1080	0,128985
32 KB	640	0,130264
64 KB	470	0,131786
128 KB	310	0,133511
256 KB	220	0,144560
512 KB	130	0,144513
1024 KB	90	0,161233
2048 KB	70	0,186473

We made time measurements for the accounting agent monitoring thread (Fig. 8). The measuring presented in the graph refers to a testing scenario, measuring the CPU time spent for delivering the service to a user (*user time*) and the time spent by the system to support the thread (*overhead time*).

The label presented in the x-axis represents the sleep time (periodicity) of the monitoring thread (accounting agent implementation)—*S0* means the monitoring thread did not sleep, processing "continuously" (according to the JVM scheduling priority scheme). On the other hand, $x \neq 0$ in Sx means the time (in milliseconds), between consecutive monitoring tasks, the thread slept (e.g., *S2* means thread slept for ~ 2 ms and *S2048* thread slept for ~ 2 s).

The threads were executed with the same priority to avoid disparities on the measuring due to scheduling priorities on the algorithm adopted by the JVM.

In Fig. 8, it is possible to observe that the experiments show the best periodicity for service accounting on the developed scenario. In this case, we can see that around *S32* (accounting monitoring periodicity of *32ms*), it achieved the best tradeoff between overhead and processing time (user time) for the three CPU loads considered in a testing scenario. Thus, this period (32 ms) defines the maximum amount of time that a given user may be under an exception condition and, therefore, the best reevaluation periodicity.

Fig. 8 shows that a high frequency monitoring may degrade the service's performance (from S0 to S8, on average, the overhead time is higher), because the monitor will use system's resources in a more intensive manner.

We executed some stress testing on tuple space (Table 1) aiming to identify the number of tuples it can store consecutively. One can notice that, as a rule, while the tuple size (consumption attributes) doubles the number of tuples stored reduces to a half. However, the time spent to store such a number of tuples varies significantly, yelding the best throughput for tuples of size between 4KB and 16KB.

In the case of the experiment reported in Fig. 8, the accounting agent sent tuples with ~ 2 KB (17 usage attributes).

An analogous test was performed on the PDP for reevaluation policies (Table 2). But, in such a case, the PDP retrieved the policy reevaluation request from tuple space with ~ 2 KB (considering a tuple header has about 1 KB) and the policy set from PAP; a simple XACML policy (defining one authorization rule) is about 4 KB.

Next, the PDP evaluates the request and posts the response back to the tuple space with ~ 2 KB (considering a tuple header has about 1 KB). Thus, the experiment results the PDP power to serve policies reevaluation requests (Table 2, "No. of parallel policies reevaluated before crashing") and the time spent to do it (Table 2, "Time spent to evaluate a policy").

We could evaluate the time spent by the PIP to get attributes from the tuples space and store it locally, which is very similar to the PDP's spent time to evaluate a policy (Table 2). Moreover, the time spent by the PEP to write an evaluation request on the tuple space is very similar to that spent by accounting agents for writing a consumption attribute. Therefore, considering the time for policy reevaluation request on the tuple space added to the time required by the PDP to evaluate and answer the request, we conclude that for many combinations of request size and policy size the total time spent to complete the process of reevaluation stays below 32 ms (Fig. 8).

Taking into account the time spent to store a consumption tuple, from accounting agent, added to the time spent by PIP to store it locally, we can conclude that these attributes will be available for the PDP to retrieve it and consider it only in the reevaluation period (after 32ms).

We can infer that the real time needed to get a consumption attribute stores it on the PIP and effectively uses it on the reevaluation of a policy will take two periods of 32ms, one to have the accounting attributes available and another to use them in the reevaluation. Thus, we are reconsidering 64ms to be the best reevaluating period.

The measurements shown in Tables 1 and 2 provide an idea about the proposal's "elasticity trigger," meaning the number of servers in the pool should be increased when the demand is close to provoking a crash on the servers. The tuple space together with the distributed evaluation scheme (many PDPs instantiated on demand) provides elasticity to the $\mathrm{UCON}_{\mathrm{ABC}}$. The space for managing attributes, responsibility of the PIP, follows the same approach, providing elasticity to the accounting system for $\mathrm{UCON}_{\mathrm{ABC}}$.

7 Conclusion

This work presented an innovative approach to reevaluation of ongoing authorization for usage control. The continuous service accounting and reevaluation of authorization attributes allowed the identification of disparities between authorizations and policies. Moreover, it provided resilience (relaxing the policy rules) to authorization attributes in some circumstances without any loss to the consumer (SLA violation). When a disparity is identified and resilience is not possible, the user will be under an exception condition. In this case, the consumer's will have some alternatives to fix the situation that is very advantageous in comparison to traditional approaches present in the literature.

The proposed accounting service and continuous reevaluation for each user provides fine-grained accounting and access control for the cloud computing environment.

The resilience applied on the proposal made the authorization attributes (quotas) defined for each consumer's user more flexible. Besides, control policy violations are monitored and treated by the management environment at the federation (SLAs) and consumer level (exception

conditions). This scheme allows the flexible usage of computational resources, tweaking the quotas without waste, idling or abuse of contracted services.

The proposal's approach showed that it is possible to perform attributes management and consolidation by using loosely coupled and open standards (e.g., web services). Moreover, it is adequate to the access level allowed nowadays on the infrastructure (IaaS), not requiring changes on this environment, as the management is made by services that work according to the consumer's demand. This means that, without the proposal, the consumer would not have usage control over the cloud as no IaaS provider offers similar service. Furthermore, we could not find similar proposals to this one on the technical literature, not even for the PaaS or SaaS.

Acknowledgments

This work was partially sponsored by the Program Center for the Research and Development on Digital Technologies and Communication (CTIC/MCTI), grant 1313 and the National Council for Scientific and Technological Development (CNPq), grants 310671/2012-4 and 478285/2011-6.

References

[1] N. Li, Q. Wang, and M. Tripunitara, "Resiliency Policies in Access Control," *ACM Trans. Information and System Security,* vol. 12, article 20, 2009.
[2] Q. Wang and N. Li, "Satisfiability and Resiliency in Workflow Authorization Systems," *ACM Trans. Information and System Security,* vol. 13, no. 4, article 40, 2010.
[3] M. Stihler, A.O. Santin, A.L. Marcon Jr., and J.S. Fraga, "Integral Federated Identity Management for Cloud Computing," *Proc. Fifth IFIP NTMS,* pp. 1-5, 2012.
[4] "Security Guidance for Critical Areas of Focus in Cloud Computing," CSA, http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf, Mar. 2013.
[5] M. Yildiz, J. Abawajy, T. Ercan, and A. Bernoth, "A Layered Security Approach for Cloud Computing Infrastructure," *Proc. Int'l Symp. Pervasive Systems, Algorithms, and Networks,* pp. 763-767, 2009.
[6] "Web Services Architecture," W3C, www.w3.org/TR/ws-arch, Mar. 2013.
[7] X. Yang, B. Nasser, M. Surridge, and S.A. Middleton, "Business-Oriented Cloud Federation Model for Real-Time Online Interactive Applications," *Future Generation Computer Systems,* vol. 28, pp. 1158-1167, 2012.
[8] M. Stihler, A.O. Santin, A. Calsavara, and A.L. Marcon Jr., "Distributed Usage Control Architecture for Business Coalitions," *Proc. IEEE 44th IEEE ICC/CISS,* pp. 1-6, 2009.
[9] J. Park and R. Sandhu, "The UCONABC Usage Control Model," *ACM Trans. Information and System Security,* vol. 7, no. 1, pp. 128-174, 2004.
[10] R. Teigão, C.A. Maziero, and A.O. Santin, "Applying a Usage Control Model in an Operating System Kernel," *J. Network and Computer Applications,* vol. 34, pp. 1342-1352, 2011.
[11] Í. Goiri, F. Julià, J.O. Fitó, M. Macías, and J. Guitart, "Supporting CPU-Based Guarantees in Cloud SLAs via Resource-Level QoS Metrics," *Future Generation Computer Systems,* vol. 28, pp. 1295-1302, 2012.
[12] S. Capizzi and A.A. Messina, "Tuple Space Service for Large Scale Infrastructures," *Proc. IEEE 17th Workshop Enabling Technologies: Infrastructure for Collaborative Enterprises,* pp. 182-187, 2009.
[13] A. Michael, F. Armando, G. Rean, D.J. Anthony, H.K. Randy, K. Andrew, L. Gunho, A. David, A.R. Patterson, and Z. Matei, "Above the Clouds: A Berkeley View of Cloud Computing," technical report, 2009.
[14] J.S. Erickson, S. Spence, M. Rhodes, D. Banks, J. Rutherford, E. Simpson, G. Belrose, and R. Perry, "Content-Centered Collaboration Spaces in the Cloud," *IEEE Internet Computing,* vol. 13, no. 5, pp. 34-42, Sept./Oct. 2009.
[15] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *NIST Information Technology Laboratory,* 2009.
[16] M. Pearce, S. Zeadally, and R. Hunt, "Virtualization: Issues, Security Threats, and Solutions," *ACM Computing Survey,* vol. 45, no. 2, article 17, 2013.
[17] B. Grobauer, T. Walloschek, and E. Stöcker, "Understanding Cloud-Computing Vulnerabilities," *IEEE Security and Privacy,* vol. 9, no. 2, pp. 50-57, Mar./Apr. 2011.
[18] OASIS, eXtensible Access Control Markup Language, www.oasis-open.org/committees/xacml, Mar. 2013.
[19] L.J. Zhang and J. Zhang, "An Integrated Service Model Approach for Enabling SOA," *IT Professional,* vol. 11, no. 5, pp. 28-33, Sept./Oct. 2009.
[20] H.C. Lim, S. Babu, J.S. Chase, and S.S. Parekh, "Automated Control in Cloud Computing: Challenges and Opportunities," *Proc. First Workshop Automated Control for Datacenters and Clouds,* pp. 13-18, 2009.
[21] S. Bertram, M. Boniface, M. Surridge, N. Briscombe, and M. Hall-May, "On-Demand Dynamic Security for Risk-Based Secure Collaboration in Clouds," *Proc. IEEE Third CLOUD,* pp. 518-525, 2010.
[22] P. Goyal and R. Mikkilineni, "Policy-Based Event-Driven Services-Oriented Architecture for Cloud Services Operation & Management," *Proc. IEEE Second CLOUD,* pp. 135-138, 2009.
[23] T. Tavizi, M. Shajari, and P.A. Dodangeh, "Usage Control Based Architecture for Cloud Environments," *Proc. IEEE 26th Int'l Parallel and Distributed Processing Symp. Workshops,* pp. 1528-1533, 2012.
[24] C. Danwei, H. Xiuli, and R. Xunyi, "Access Control of Cloud Service Based on UCON," *Proc. First Int'l Conf. Cloud Computing,* pp. 559-564, 2009.
[25] N. Amit, C. Sanjay, and S. Gaurav, "Policy Based Resource Allocation in IaaS Cloud," *Future Generation Computer Systems,* vol. 28, pp. 94-103, 2011.
[26] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification - RFC 2093 and Architecture - RFC 2094," 1997.
[27] W3C, SOAP Version 1.2., www.w3.org/TR/soap, Mar. 2013.
[28] OASIS, Web Services Security SOAP Message Security 1.1., http://docs.oasis-open.org/wss/v1.1, Mar. 2013.
[29] OASIS, WS-Trust 1.4, http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/, Mar. 2013.

Arlindo Luis Marcon Jr. received the BS degree in informatics from União da Vitória College in 2004, the MSc degree in 2008 and the PhD degree in 2012 in computing science from the Pontifical Catholic University of Paraná. He has been an associated professor at the Federal Institute of Paraná, PR, Brazil, since 2010. His main research interests include security, access control, distributed systems, web services, and cloud computing.

Altair Olivo Santin received the BS degree in computer engineering from the Pontifical Catholic University of Paraná in 1992, the MSc degree in electrical engineering and industrial computer from the Technological Federal University of Paraná in 1996, and the PhD degree in electrical engineering from the Federal University of Santa Catarina, Brazil, in 2004. He is a full professor of computer science at the Pontifical Catholic University of Paraná. His research interests in computer security include usage and access control models and mechanisms for distributed systems, web services and cloud computing security, intrusion detection systems and digital forensics. He is a member of the IEEE, ACM, and the Brazilian Computer Society.

Maicon Stihler received the BS degree in information systems from Unidavi College in 2004 and the MSc degree in computer science from the Pontifical University of Paraná in 2009. He is currently working toward the PhD degree in Graduate Program of Computer Science at the Pontifical University of Paraná. His main research interests include computer security, usage control, distributed systems, and cloud computing.

Juliana Bachtold Jr. received the BS degree in computer science from the University of Santa Catarina State in 2002 and the MSc degree in computing science from the Pontifical Catholic University of Paraná in 2012. Her main research interests include information technology management, information security, and computer network.