Ontology-based Negotiation of Security Requirements in Cloud

Loredana Liccardo, Massimiliano Rak
Dipartimento di Ingegneria dell'Informazione
Seconda Università di Napoli
{*loredana.liccardo, massimiliano.rak*}@*unina2.it*

Giuseppe Di Modica, Orazio Tomarchio
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica
Università di Catania
{*Giuseppe.DiModica, Orazio.Tomarchio*}@*dieei.unict.it*

***Abstract*—The Cloud Computing paradigm attracts many customers because of the potentialities it promises. Despite of many benefits, a widespread adoption is limited by many issues that potential customers still have to face. Security in the cloud is one of the main concern for the customer. The Cloud Service Provider (CSP) is responsible of providing security to customers and assuring that their data and application are properly secured. In this context, the concept of Service Level Agreement (SLA) assumes a great importance. It can be used as a means to formalize and establish in a contract what must effectively be granted in terms of security levels. There is actually *Semantic Gap* between how security guarantees are intended respectively by customers and providers. A customer is inclined to express security in terms of high-level requirements, while a CSP expresses guarantees through a technical, low-level language. To address this gap, the key is to find a common language for both the customer and the CSP. The goal of this paper is to offer an Ontology-based Negotiation Service allowing a customer to negotiate the interested security level among different CSPs, with the possibility to choose the best security offering; a Security Ontology was developed as a basis for a common semantic language that customers and providers will have to use to express security features and requirements.**

***Keywords*-Cloud computing; Security; SLA; Negotiation; Ontology;**

I. Introduction

Cloud Computing has emerged as a computing paradigm able to offer resources in a flexible, dynamic, resilient and cost-effective way. Despite the potentialities promised to Customers are very attractive, a full cloud adoption is hindered because of many issues and challenges posed by the cloud model. Specifically, security was identified as one of the major obstacle to the adoption of cloud services. Being the cloud model based on an on-demand delivery of computing resources through Internet, accessed in a self-service way, the control over physical resources is no longer in the hands of the Customer but in those of the Cloud Service Provider (CSP). For this reason, CSPs are responsible of providing security to Customers and assuring that Customers' data and applications are properly secured.

In this respect the concept of Service Level Agreement (SLA) assumes a considerable importance. An SLA is an agreement between a CSP and a Customer, which describes the Service and the associated quality levels, and specifies the responsibilities of both CSP and Customer. It is a means to formalize, establish and guarantee by means of a signed contract what is effectively granted in terms of *Quality of Service*.

One of the major issues regarding the definition of security guarantees in a SLA is the one we defined *Semantic Gap*. On the one side, a Customer, who is interested in the purchase and the use of a service, is prone to express security in terms of high-level requirements, while on the other side, a CSP is inclined to express their own guarantees in terms of low level security mechanisms and algorithms. The result is a hard mutual understanding process which produces the perception of a poor security, particularly from the point of view of the Customer.

In order to address this gap, finding a common language to be "spoken" by both the Customer and the CSP is absolutely necessary. Many have tried to address, often in a non-exhaustive way, the definition and characterization of security requirements [1]. An interesting approach is to use the "smart notices" [2] to produce a synthetic version of the SLA for the Customer and multiple representations of the SLA.

Our proposal is focused on a *user-centric* semantic approach, and uses a Security Ontology as a key for a common language. On the Customer side, the concepts defined by the security ontology are used to express the Guarantee Terms of the SLA. On the CSP side, appropriate technical languages are used. The final goal of the paper is to offer a semantic negotiation service that leverages on such a protocol to enable the sign of a SLA on security parameters.

The structure of the paper is organized as follows: section II focuses on the problem formalization and illustrates the proposed approach. Section III presents the semantic matching technique. Section IV depicts the Ontology-based Negotiation Model. Section V discusses a concrete case study and finally section VI concludes the work.

II. Problem Formalization and Proposed approach

In this section we focus on the formalization of the problem by describing the requirements of the application and the adopted approach.

As stated in the Introduction, the goal is to offer an Ontology-based Negotiation Service allowing a Customer

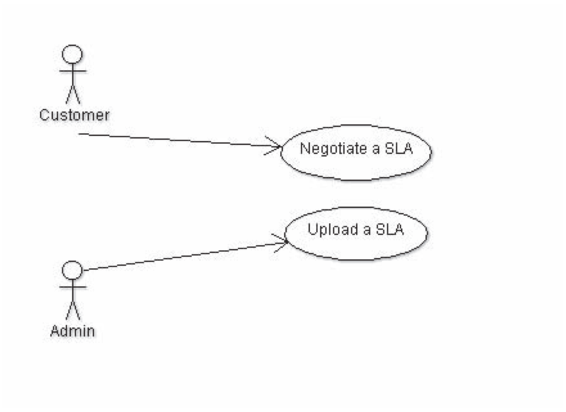

Figure 1. Users of the Ontology-based Negotiation Service

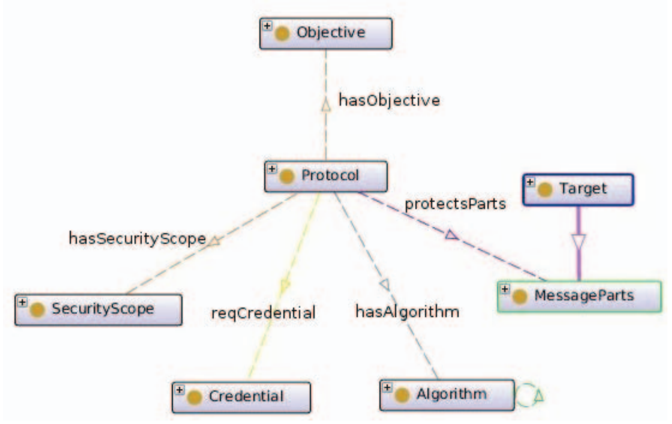

Figure 2. Security ontology

to negotiate the involved security parameters with different Cloud Providers, also enabling to choose the best security offering. In this scenario, there are two types of users that make use of the service, as depicted in Figure 1: (i) a *Customer*, who wants to negotiate a security level of a cloud service and (ii) a *Cloud Administrator* of a Cloud infrastructure (from now *Admin*), who defines the security offering of a CSP and uploads it in a repository used by our negotiation service.

The Service Level Agreement (SLA) is the means to reach an agreement between a Customer and a CSP. It allows to establish security levels. In our approach we make use of the security concepts defined in the Security Ontology to build up the Guarantee terms of the SLA. The core idea is to use an ontology as a common ground to express the security concepts, in such a way to address the semantic gap between the Customer and the CSP.

The solution we propose can be summarized as follows:

- it is based on an intelligent negotiation service to which a Customer and an Admin delegate the charge of reaching an agreement.
- the negotiation service presents two interfaces: one for the Customer and one for the Admin. On the one hand, the Admin of a CSP publishes their own offerings expressed in a formal and specific language. On the other hand, the Customer submits a SLA request, expressed in a high-level language, based on a security ontology, and receives counter proposals based on the same ontology. The overall negotiation process will be thoroughly described in section IV.

The problem we address in this paper is how to manage the matching of the two languages: the Customer's and the Admin's.

III. Semantic matchmaking

An Admin will advertise the security features of the cloud infrastructure they manage by means of *security policies*. Security policies express respectively security requirements and capabilities of a CSP. Requirements represent constrains to Customers who wish to use the provided services, while capabilities are the security level that the CSP is capable to offer to Customers. Likewise, in the proposed model, Customers will have to express their needs in the form of security requirements and capabilities.

When the Negotiation takes place (see section IV), the Customer's and the CSP's policies are compared by overlapping the requirements defined in one policy on the capabilities expressed in the other, and vice versa. This way every possible requirement-capability pair is assigned a match level and, in the end, the overall match level between the Customer and the CSP is obtained.

The specification of the security policy is a semantic extension of WS-Policy [3]. Since WS-Policy does not impose any limit on the kind of allowed assertions, we decided to adopt semantically enriched terms. To this purpose we have defined a **Security Ontology**, which is an ontology that describes main concepts from the security domain. The Security Ontology is a non-exhaustive specification of concepts from the security domain. It is extensible, in the sense that the missing concepts can be easily included.

Key concepts for our ontology are the *security objectives* and the *security protocols*. Figure 2 is a comprehensive picture of the ontology's base concepts. For space reason, the rest of the concepts were not depicted.

A *security objective* is an abstraction of the type of protection that must be enforced. Examples are confidentiality, integrity, non-repudiation and authentication. A *protocol* is the set of rules used for the communication among two or more entities. It encompasses semantic individuals such as XML-Encoding, XML-DigitalSignature, SAML, XACML and HTTPS. A protocol can be bound to an *Algorithm*, which can be one out of cipher, integrity, key wrapping or key derivation type. Protocols can also be bound to *Credential*, which for instance must be provided in the context of an authentication as identity proof. We can distinguish among physical, biometric or electronic credentials. *SecurityScope* refers to scope of the security, and can be either machine-bound (firewall) or communication-bound (network level, transport level, message level). Finally, *Target* is a generic concept and can comprise several other concepts. Currently it is subclassed by just the *MessageParts* concept, whose elements are those that can be found within a message.

Here is an example of what a semantic security policy may look like:

```
<?xmlversion="1.0"encoding="UTF-8"?>
<wsp:Policy xmlns:rdf="http://www.w3.org/1999/02/22
   -rd-syntax-ns#" .............

<wsp:ExactlyOne>
  <wsp:All>

    <security:LoginProtocol rdf:ID="requirement0" >
      <security:requiresCredential
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#OneTimePassword"
security:isMainCredential="false"/>
    </security:LoginProtocol>

    <security:Https rdf:ID="requirement1" >
      <security:hasSecurityScope
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#TransportLevelSecurity"/>
      <security:requiresCredential
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#Timestamp"/>
      <security:hasEncryptionAlgorithm
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#AES-256"/>
      <security:hasSignatureAlgorithm
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#SHA-1"/>
    </security:Https>

  </wsp:All>
</wsp:ExactlyOne>

</wsp:Policy>
```

Let us then assume that we need to match two policies representing requirements and capabilities of, respectively, a Customer and of a CSP. The process of making the match between the two policies consists in searching a correspondence between requirements of a party and capabilities the other party. Specifically, a) the requirements of a CSP are compared to the capabilities of the Customer, and b) the capabilities of the CSP are compared to the requirements of the Customer. In order for the comparison process to have a positive outcome, two conditions must hold:

- the capabilities advertised in the CSP's policy must meet the requirements expressed in the Customer's policy;
- the requirements expressed in the CSP's policy must be met by the capabilities advertised in the Customer's policy.

The matchmaking process breaks down into two steps: assigning the match level to each requirement-capability pair and assigning the match level between to overall set of requirements and the overall set of capabilities.

As for the first step, each requirement-capability pair can be assigned one out of four different match levels:

- Perfect Match;
- Close Match;
- Possible Match;
- No Match;

For each requirement, the objective is to find the capability that matches at best. In the second step the overall match between the two policies will be evaluated. The overall match is defined to be the minimum among the individual match levels evaluated in the first step for each requirement-capability pair. The reader may find more details on the matchmaking process in [4].

IV. NEGOTIATION PROCESS

As described in Section II, the goal of our approach is to offer a service which is able to support the negotiation between Customers and CSPs by leveraging on the security ontology above presented as a common language. The typical usage of such a service is the resource brokering: Customers aim at acquiring resources (Virtual Machines,..) from different CSPs, each offering different levels of guarantee in term of security and QoS in general.

In order to manage the negotiation process between Customers and CSPs we propose the architecture shown in Figure 3. Such a solution assumes that Customers negotiate with a third, independent party the quality of the services offered by CSPs. Such Third party (named *Broker*) is able to take into account offerings from different CSPs, uploaded in the system by Admins of several CSP as shown in the picture, and helps the Customer to make a choice about the one that better fits their need.

As shown in the picture, the Third party's architecture is made of three different modules: the *Negotiation Module*, which has the role of managing the interactions with Customers, the *Service Brokering*, which is in charge of brokering the individual services to be offered, and the *Policy Matching*, which evaluates the security levels by matchmaking the security policies contained in the SLAs.

Security parameters are often independent from the individual service offered, but instead, they are related to the overall offering of a single provider. This implies that negotiation of security parameters can be done independently of the individual service the Customer is interested on, while other features (i.e. performances) should be negotiated each time a new service is requested. Therefore, making this assumption, our Negotiation Model performs a two-phase negotiation. In the first phase Customers negotiate the security level, independently of the service they wish to request; once the negotiation has identified the providers that best fit the security needs of the Customer, the specific service is negotiated.

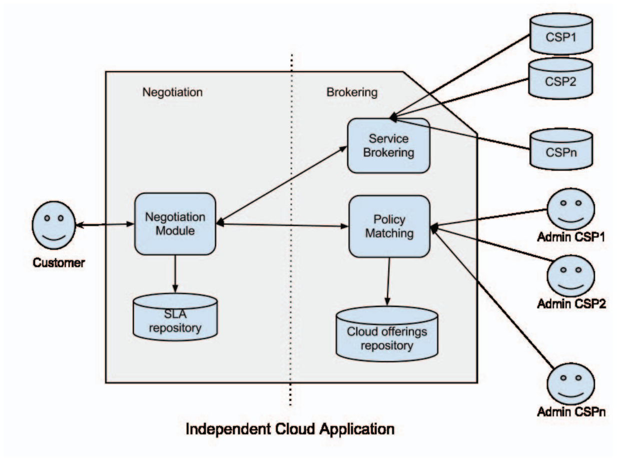

Figure 3. Architecture of a Third Party Negotiating Service

In this paper we will focus on the first phase of such a negotiation, while the second phase is fully described in [5], that is based on the idea presented in [6]. In the following, we will propose the general purpose negotiation protocol adopted by the Negotiation Module.

The goal is to offer Customers a simple application which automates the brokering procedure and takes into account security parameters as well as other functional and non-functional requirements.

A. Negotiation Protocol

The Negotiation Protocol on which our solution is built is a general purpose protocol. In this paper, we use it to negotiate just security levels.

As mentioned in Section II, there are two parties: a Customer who wants to negotiate a security level for a service, and an Admin who uploads the security offering of a CSP in a repository that is available to the negotiation application.

The offering of an Admin can be described in any language more near to the Admin skills, but we consider that an offering is expressed by using the WS-Policy specification. This document contains concepts defined in a security ontology.

The proposed Negotiation Protocol presents the following requirements:

- a Customer describes their security requirements by means of an XML document is annotated with concepts defined in a security ontology; such document is then inserted into a WS-Agreement compliant document through a more high-level mechanism, in a way that is absolutely transparent to the Customer. Then the Customer submits this proposal to an automatic system,
- the automatic system should be able to return, without any human interaction, a minimal set of CSPs whose offers best fit the user needs. In the case that only one offer is returned, an automatic brokering is performed; otherwise the Customer is asked to choose.

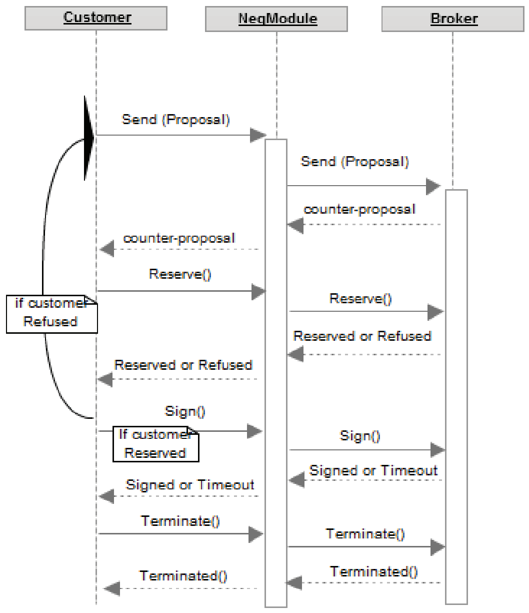

Figure 4. Negotiation Protocol

The negotiation protocol is illustrated in Figure 4 through a sequence diagram.In this general negotiation protocol, the entities involved are the Customer, the Negotiation Module and the Broker, which can be either the Policy Matching or the Service Brokering.

In the following, we discuss the negotiation protocol considering as Broker the Policy Matching. The Customer, after filling an SLA proposal, starts up the negotiation by submitting the SLA to the Negotiation Module. The latter sends it to the Policy Matching that, making use of the semantic matching described in Section III, performs a SLA evaluation and comparison between the Customer SLA proposal and the CSP offerings. Specifically, through this semantic approach, one of different match levels is assigned to each CSP offering.

As a result, the Policy Matching returns to the Customer an ordered set of CSP offerings closer to their original request. The Customer can either choose to reserve a certain SLA or to refuse it. In case the reserved SLA is rejected, the Customer can submit a new proposal, continuing the negotiation; instead, if the SLA gets successfully reserved, a timeout is set for the SLA, within which the Customer will have to formally sign the agreement. According to the WS-Agreement protocol, the Customer is free to terminate the agreement anytime after the contract sign.

V. An Example

In this section we describe a simple example of application of the proposed solution. As Cloud Service Provider we consider the PerfCloud solution [7], that, providing a set of services for the creation of Virtual Clusters, offers an Infrastructure as a Service (IaaS). As explained in [8], we consider that the PerfCloud CSP offers a set of services enforcing different security policies. We assume that the Customer, through our Ontology-based Negotiation Service, negotiates the security level with PerfCloud. PerfCloud security offers focus on user's authentication and authorization. In particular, [8] provides a description of the security parameters on which PerfCloud is based. On the other side, the Customer specifies their security requirements in a WS-Agreement language. In the following we show an example of the Customer's document. In particular, we show an excerpt of the whole document related to the "Guarantee Term" section. It consists of two Guarantee Terms: one is related to the authentication feature (*WSAuthLevel*) and one to the authorization feature (*WSAuthZLevel*). As for the authentication, a Customer can choose one out of SecureMessage, SecureConversation, SecureTransport, offered by PerfCloud; whereas for authorization an XACML mechanism is used by the CSP. As shown in the example below, the Customer selects SecureMessage for the authentication method and XACML for the authorization. These represent security requirements for the Customer. We assume that our Ontology-based Negotiation Service, once received this document, translates it into the WS-Policy format.

```
<wsag:GuaranteeTerm wsag:Name="TransportLevel">
  <wsag:ServiceScope wsag:ServiceName="perfCloudVbox"/>
  <wsag:QualifyingCondition>SERVICE_STATE eq 'Ready'
  </wsag:QualifyingCondition>
  <wsag:ServiceLevelObjective>
      <wsag:KPITarget>
          <wsag:KPIName>TransportLevel</wsag:KPIName>
          <wsag:CustomServiceLevel>Transport eq HTTP
          </wsag:CustomServiceLevel>
      </wsag:KPITarget>
  </wsag:ServiceLevelObjective>

 <wsag:BusinessValueList>
        <wsag:Penalty>
            <wsag:AssessmentInterval>
                <wsag:TimeInterval>Always
                </wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:Count>3</wsag:Count>
            <wsag:ValueExpression>DisableAccount
            </wsag:ValueExpression>
        </wsag:Penalty>
    </wsag:BusinessValueList>
</wsag:GuaranteeTerm>

<wsag:GuaranteeTerm wsag:Name="WSAuthLevel">
    <wsag:ServiceScope
    wsag:ServiceName="perfCloudVbox"/>
    <wsag:QualifyingCondition>SERVICE_STATE eq 'Ready'
    </wsag:QualifyingCondition>
    <wsag:ServiceLevelObjective>
        <wsag:KPITarget>
            <wsag:KPIName>WSAuthLevel</wsag:KPIName>
            <wsag:CustomServiceLevel>WSAuth eq
            SecureMessage
            </wsag:CustomServiceLevel>
        </wsag:KPITarget>
    </wsag:ServiceLevelObjective>

<wsag:BusinessValueList>
        <wsag:Penalty>
            <wsag:AssessmentInterval>
                <wsag:TimeInterval>Always
                </wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:Count>3</wsag:Count>
            <wsag:ValueExpression>DisableAccount
            </wsag:ValueExpression>
        </wsag:Penalty>
    </wsag:BusinessValueList>
</wsag:GuaranteeTerm>

<wsag:GuaranteeTerm wsag:Name="WSAuthZLevel">
    <wsag:ServiceScope
    wsag:ServiceName="perfCloudVbox"/>
    <wsag:QualifyingCondition>
    SERVICE_STATE eq 'Ready'
    </wsag:QualifyingCondition>
    <wsag:ServiceLevelObjective>
        <wsag:KPITarget>
            <wsag:KPIName>WSAuthZLevel
            </wsag:KPIName>
            <wsag:CustomServiceLevel>
            WSAuthZ eq XACML
            </wsag:CustomServiceLevel>
        </wsag:KPITarget>
    </wsag:ServiceLevelObjective>

<wsag:BusinessValueList>
        <wsag:Penalty>
            <wsag:AssessmentInterval>
                <wsag:TimeInterval>Always
                </wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:Count>3</wsag:Count>
            <wsag:ValueExpression>DisableAccount
            </wsag:ValueExpression>
        </wsag:Penalty>
    </wsag:BusinessValueList>
</wsag:GuaranteeTerm>
</ws:Terms>
<ws:CreationConstraints/>
```

We assume that the Admin of the CSP uploads on the system three offers based on three different security levels; in the following we consider just one example WS-Policy offer of the PerfCloud. It is the offer for which the matching process will return a *Perfect Match*. As we can see, it is composed of two capabilities provided by the CSP: one related to SecureMessage and one related to XACML.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<wsp:Policy xmlns:rdf="http://www.w3.org/1999/02/22
-rd-syntax-ns#" .............

<wsp:ExactlyOne>
<wsp:All>

<security:MessageLevelSecurity rdf:ID="capability0" >
<security:requiresCredential
rdf:resource="http://www.semanticweb.org/ontologies/
security.owl#Certificate"/>
</security:MessageLevelSecurity>
```

```
<security:XACML rdf:ID="capability1" >
</security:XACML>

</wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
```

VI. Conclusion and Future Works

In this paper we have discussed about the security issues that hinder the adoption of cloud services and, as a consequence, how the concept of SLA assumes a considerable importance in the cloud environment. SLA is a means used to formalize, establish and to guarantee in a contract what is effectively granted in terms of security levels. We have addressed the *Semantic Gap* regarding how security guarantees are intended by respectively Customers and Cloud Service Providers (CSP). We believe that a common language shared between the Customer and the CSP is the key to address this issue. To this end, we have defined a Security Ontology to be used by Customers and CSPs to reciprocally communicate each other's security requirements. The work presented in the paper proposes an Ontology-based Negotiation Service allowing a Customer to negotiate the needed security level among different CSPs, with the possibility to choose the security offering that best suites the Customer's need. In the future, we will extend our work by providing the Customer with a more user-friendly mechanism, based on a a questionnaire, that will guide them to specify security requirements by just answering a set of easily understandable questions. The outcome of this process will be the WS-Agreement document to submit to the Ontology-based Negotiation Service.

Acknowledgment

This work has been supported by the FP7-ICT-2009-5-256910 (mOSAIC) EU project and by the MIUR-PRIN 2008 Cloud@Home: a New Enhanced Computing Paradigm.

References

[1] P. H. Meland, K. Bernsmed, M. G. Jaatun, A. Undheim, and H. Castejon, "Expressing cloud security requirements in deontic contract languages," in *CLOSER*, 2012, pp. 638–646.

[2] S. Pearson and P. Tsiavos, "From creative commons to smart notices - designing user centric consent management systems for the cloud," in *CLOSER*, 2012, pp. 647–660.

[3] "Ws-policy specification, web services policy framework," September 2004.

[4] G. Di Modica and O. Tomarchio, "Semantic security policy matching in service oriented architectures," in *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, 2011, pp. 399–405.

[5] A. Amato, L. Liccardo, M. Rak, and S. Venticinque, "Sla negotiation and brokering for sky computing," 2012, pp. 611–620, cited By (since 1996) 0.

[6] S. Venticinque, R. Aversa, B. Di Martino, M. Rak, and D. Petcu, "A cloud agency for sla negotiation and management," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6586 LNCS, pp. 587–594, 2011, cited By (since 1996) 1.

[7] V. Casola, M. Rak, and U. Villano, "Perfcloud: Performance-oriented integration of cloud and grid," in *Cloud Computing*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2010, vol. 34, pp. 93–102.

[8] M. Rak, L. Liccardo, and R. Aversa, "A sla-based interface for secure authentication negotiation in cloud," *Journal of Information Assurance and Security*, vol. 2, no. 7, pp. 137–146, 2012.