Expressing cloud security requirements for SLAs in deontic contract languages for cloud brokers

Per Håkon Meland*, Karin Bernsmed and Martin Gilje Jaatun

Department of Software Engineering, Safety and Security,
SINTEF ICT,
Strindveien 4, N-7465 Trondheim, Norway
E-mail: per.h.meland@sintef.no
E-mail: Karin.Bernsmed@sintef.no
E-mail: martin.g.jaatun@sintef.no
*Corresponding author

Humberto Nicolás Castejón and Astrid Undheim

Telenor Research and Future Studies,
Otto Nielsens veg 12, N-7052 Trondheim, Norway
E-mail: humberto.castejon@telenor.com
E-mail: astrid.undheim@telenor.com

Abstract: The uptake of cloud computing is hindered by the fact that current cloud SLAs are not written in machine-readable language, and also fail to cover security requirements. This article considers a cloud brokering model that helps negotiate and establish SLAs between customers and providers. This broker handles security requirements on two different levels: between the customer and the broker, where the requirements are stated in natural language; and between the broker and different cloud providers, where requirements are stated in deontic contract languages. There are several such languages available today with different properties and abstraction levels, from generic container languages to more domain-specific languages for specifying the various details in a contract. In this article, we investigate the suitability of ten deontic contract languages for expressing security requirements in SLAs, and exemplify their use in the cloud brokering model through a practical use case for a video streaming service.

Keywords: cloud computing; security; requirements; contracts; service level agreements; SLAs; brokering.

Reference to this paper should be made as follows: Meland, P.H., Bernsmed, K., Jaatun, M.G., Castejón, H.N. and Undheim, A. (2014) ‘Expressing cloud security requirements for SLAs in deontic contract languages for cloud brokers’, *Int. J. Cloud Computing*, Vol. 3, No. 1, pp.69–93.

Biographical notes: Per Håkon Meland received his MS in Computer Science in 2002 from the Norwegian University of Science and Technology (NTNU). Since then he has been working for SINTEF ICT and currently holds the position of Senior Research Scientist. His research interests include secure software engineering and composition of dynamic services. He is a member of the Association for Computing Machinery (ACM).

Karin Bernsmed is a Research Scientist at SINTEF ICT. Her research interests include, amongst other things, security and privacy in cloud computing.

Martin Gilje Jaatun is a Senior Scientist at SINTEF ICT, where he has been employed since 2004. His research interests include security in cloud computing and critical information infrastructures. He is the Vice Chairman of the Cloud Computing Association (cloudcom.org) and a senior member of the IEEE.

Humberto Nicolás Castejón is a Research Scientist at Telenor Research and Future Studies. His research interests include specification and model analysis methods of distributed, reactive systems, as well as SLA-driven resource allocation mechanisms in cloud computing.

Astrid Undheim is a Research Scientist at Telenor Research and Future Studies. Her research interests include performance and dependability modelling and analysis, lately focusing on QoS and SLAs in cloud computing.

This paper is a revised and expanded version of a paper entitled 'Expressing cloud security requirements in deontic contract languages' presented at 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012) held in Porto, Portugal, April 19, 2012.

1 Introduction

Cloud computing has found its way into the IT service delivery model for many of today's businesses. Recent numbers show that 10% of current enterprise application software is running in the cloud, and a 50% increase is expected within the next four years (Gartner, 2011). As pointed out by NIST (Liu et al., 2011), the continued growth will increase the number of actors in the field, making the integration of cloud services too complex to manage for the ordinary cloud customer. The *cloud broker* represents a promising and ambitious approach to manage cloud services in the near future. Its main purpose is to simplify the usage, performance and delivery of cloud services, and to help negotiate the relationships between providers and customers. These relationships are regulated through *service level agreements* (SLAs), which have become an important part of the cloud service delivery model. An SLA is a binding agreement between the cloud customer and the cloud service provider, the primary purpose of which is to state the obligations of the provider, together with the penalties if the provider fails to uphold the conditions.

Even though service availability is considered to be a critical issue, the number one barrier against adopting cloud computing services is the lack of assurance for the safekeeping of data and applications deployed in the cloud (ENISA, 2009a). Keeping data and applications in-house is considered to be less risky since the organisation has better control of the infrastructure and configuration. For a cloud customer to perform a proper risk analysis he will need to consider, for instance, where his data and applications will be stored, whether they will be encrypted at rest and in transit, and whether international standards such as ISO/IEC 27002 (2005) are being adhered to. Even though most public cloud providers are happy to answer these types of questions whenever asked, they will rarely or never make any promises regarding security in their SLAs. Rather, their SLAs are standard contracts made to fit as many customers as possible, and the providers accept little responsibility in case of a security incident or service outage. This is a major drawback for any potential customer who needs to ensure that, e.g., privacy legislation or internal security policies are conformed to. We believe that in order for cloud computing to reach its full potential as a mainstream outsourcing alternative, the customers' security requirements must be guaranteed through SLAs.

Existing SLAs cover traditional QoS requirements such as service performance and availability, as well as specification of reporting and violation handling, and are written in natural language. As pointed out by Dwivedi and Padmanabhuni (2008), security requirements are only represented through policies and not enforced through SLAs since they are considered more difficult to measure and quantify compared to other QoS requirements. Minimal work has been done on security SLA representation and enactment. This article considers a cloud broker model where the broker helps to negotiate and establish SLAs between cloud customers and cloud service providers. The focus is on security requirements, which are handled on two different levels: between the customer and the broker, where the requirements are stated in natural language; and between the broker and the different service providers, where requirements are stated in a *deontic contract language*, that is, "a language that can express the rights and obligations of parties to a contract in a form that can be parsed by software applications and processed with other data to determine state information about matters governed by the contract" (Leff and Meyer, 2007). We investigate the suitability of ten existing deontic contract languages to express security requirements in the cloud brokering model, and exemplify our results with help of a video streaming service use case.

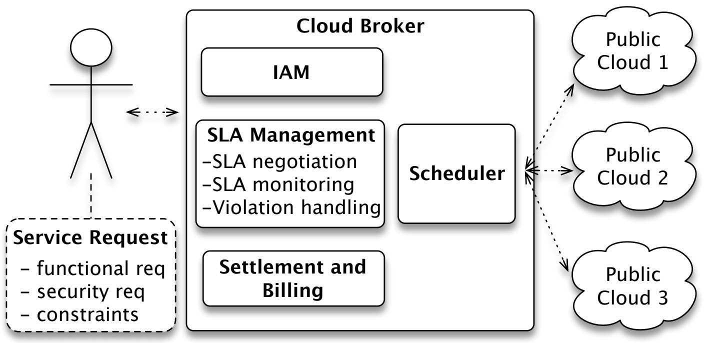
The article is organised as follows. In Section 2 we discuss the cloud service brokering model in more detail, and in Section 3 we present a use case that utilises that model to negotiate security requirements. Section 4 shows some of the deontic contract language dialects that can be used to specify security requirements in SLAs, and we compare these approaches in Section 5. Section 6 presents related work on contract languages and SLA frameworks for cloud computing, while Section 7 proposes a solution based on our case study experiences. We conclude our work in Section 8.

2 What is cloud service brokering?

In a recent Forrester report (Ried, 2011) it is argued that while reduced cost was originally the main goal for most companies adopting cloud computing, companies now see agility and flexibility as more important factors. Cloud brokers will have a vital role in providing this flexibility in the future cloud ecosystem. They will act as one-stop shops for consumers, SMEs and enterprises that look for a complete cloud service portfolio covering all their IT needs. It is also expected that cloud brokers will help reducing the perceived risk associated with consuming cloud services from different providers, as mentioned by Gartner in a recent report (Heiser and Cearley, 2011). Especially, offering single-sign on (SSO) functionality for identity federation across different cloud providers is seen as valuable. Brokering services will prove more and more appealing to SMEs, while large enterprises may perform the cloud brokering themselves (e.g., brokering between their private cloud infrastructure and public clouds for cloud bursting).

Forrester distinguishes between simple brokers that support dynamic sourcing within one cloud segment, such as public IaaS, and full brokers that provide dynamic sourcing across multiple private, hybrid and public clouds, and that offer a range of IaaS, PaaS and SaaS services. Some cloud brokering services are already commercially available (Lheureux and Plummer, 2011). Most of them, however, offer limited functionality (e.g., billing or identity management) and few of them, if any, have the capability of dynamically sourcing across different clouds. In academia, Tordsson et al. (2012) have proposed an IaaS cloud broker mechanism intended to provide cloud users with the requested number of virtual machines from multiple providers with the best cost/performance ratio, according to a given budget. Their focus is on the total infrastructure capacity and price, but they do not mention security requirements as possible constraints when selecting cloud providers. In an earlier work, Elmroth et al. (2009) discuss financial aspects of federated clouds, but they provide few details on how this would influence the SLA of the cloud customer.

Figure 1 A cloud broker model for ensuring secure cloud deployment



In this article we envision a cloud broker model as the one depicted in Figure 1. This broker will have business relationships with a set of cloud providers offering different types of services, and will act as an intermediary between them and the customers. Automatically managing SLAs with the different cloud providers on behalf of the customers will be one of the main tasks of the broker. However, there still remain quite a few challenges related to contract scenarios to make this possible, as presented by Leff and Meyer (2007):

- Currently, contract documents are created using word processing applications. These documents cannot easily be processed at convenient levels of granularity by automated systems.
- Consumers cannot easily compare terms offered by different providers.

- Contract negotiation is slow and expensive. The inefficiencies inherent in human contract negotiation limit the value of the transaction, particularly where rich parameter sets are involved.
- There is no standard way to map a given set of negotiated contract parameters to a unique set of contract terms.

The challenges above could be solved by expressing customer requirements using a suitable deontic contract language. However, few customers would have the necessary skills or technical expertise to write a deontic contract that can be automatically processed, and technical contract writing tools have not been widely accepted (Finnegan et al., 2007). Another important task of the cloud broker will therefore be to translate the customer requirements into a selected deontic contract language. This will facilitate dynamic and automatic SLA negotiation between the cloud broker and the cloud service providers, including renegotiation of SLAs or migration of service components in case of changing requirements from the users as well as measured and observed SLA violations. Since there is currently not a prevalent standard deontic contract language, the cloud broker would in many cases need to be multilingual (i.e., able to translate into several languages). In the following we look at the main broker tasks in more detail.

2.1 SLA management

According to TeleManagement Forum (2005), the SLA lifecycle consists of six distinct phases:

- *Service and SLA template development.* The service to be provisioned is developed and one or more SLA templates are created. Each SLA template describes which quality level the service is offered at, its price, penalties in case of quality level violation and other service parameters.
- *Service discovery and SLA negotiation.* Customers discover service providers and negotiate specific values for the service parameters defined in the SLA template. A successful negotiation ends up with the creation of an SLA.
- *Service provisioning.* The service, or an instance of it, is prepared for consumption by the customer. This includes the configuration of the service and the resources it uses in order to meet the SLA parameters.
- *Service execution.* The service is executed and monitored. The service quality is validated and SLA violations are handled.
- *SLA assessment.* SLAs are examined to determine if they still fit the business needs.
- *Service termination and decommission.* The service is terminated, either successfully or due to SLA violations, and the service provider undertakes decommissioning actions.

The cloud broker is called to play an important role in two of the aforementioned phases: the service discovery and SLA negotiation phase; and the service execution phase.

It is expected that the broker will negotiate and sign an SLA with a set of selected cloud providers before including their services in its own service portfolio. However, some SLA requirements and corresponding metrics may still be open for negotiation with individual customers. The requirements expressed by the individual customers may then be used by the broker's SLA management unit to discover cloud providers fulfilling those requirements. In the case of IaaS providers, the broker's scheduler will use the customer requirements to calculate the most optimal service deployment across the discovered providers, while minimising the cost for the customer. The objective is to reduce the cost of the deployment while adhering to the requirements. In the case of SaaS and PaaS providers, functional requirements will also be considered to select the most suitable providers.

After SLA negotiation, the broker will make available the requested services from the selected providers. In the case of IaaS providers, the broker would have adapters for the different infrastructure services, as described in the paper by Tordsson et al. (2012). For SaaS providers, the cloud broker could migrate data between different storage infrastructures, and also provide integration facilities for exchange of data and information between different cloud applications and cloud providers.

Once the services are being consumed, the broker will monitor the service execution and measure the QoS actually experienced. In case of SLA breaches, the broker will take actions to handle the SLA violation (e.g., request compensation), thus sparing the customer the administrative burden of doing so. Moreover, the cloud broker may provide smart SLA management, taking actions to prevent or remedy the SLA violation, such as migrating workloads from one provider to another, increasing or decreasing the level of redundancy according to the achieved availability.

2.2 End-user requirements elicitation

Ultimately, requirements in an SLA originate from an end-user – the user may have a notion of how important availability of a solution is, what kind of throughput is desired, and so on. Security and privacy requirements should also come from end-users, but unfortunately this is often difficult to achieve (Tøndel et al., 2011). In particular, users as private citizens often demonstrate the *privacy paradox*, where their actions fail to live up to the strict privacy standards they claim to aspire to.

In an enterprise setting, the organisation may have a security policy that more easily can be mapped to a set of security requirements. However, most security policies in use today do not take new computing paradigms such as cloud computing into account, and may not provide sufficient detail.

One option may be for users (or organisations) to define their default security level on a coarse scale, e.g., using a slider to select between security levels as done within the *Microsoft Internet Explorer*^{TM} internet options. When approaching a broker to request a new service, the user should then be offered the opportunity to modify individual security requirements related to the specific type of service requested. In essence, this requires the deployment of a security requirements definition tool that through a series of questions or checklists elicits requirements from the user.

A checklist approach for end-user security requirements facilitates an automatic, pre-defined translation from natural-language specification to machine-readable representation. While it would be possible for users to specify security requirements in a free-text form, this would preclude automatic processing by the broker, requiring a human-in-the-loop.

3 Use case: CloudyFilms

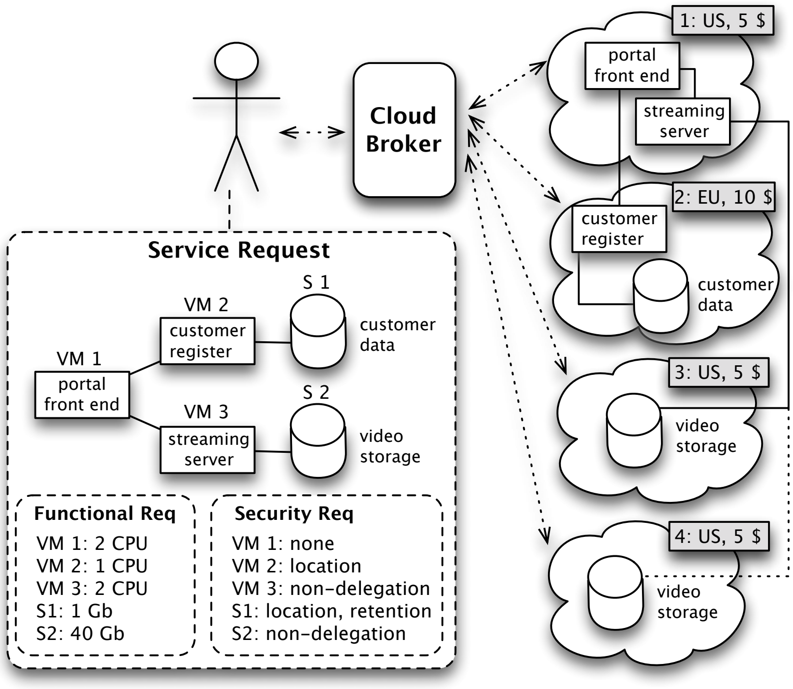
Let us consider an imaginary use case that includes a customer, a broker and several cloud service providers. The customer is a new startup company, CloudyFilms, which is planning to offer video streaming on demand with a pay-per-view business model. In order to minimise investment costs in hardware infrastructure and increase business agility, CloudyFilms decides to deploy its service in the cloud. For that, CloudyFilms needs different types of virtual resources, including computation and storage, with different functional, QoS and security requirements. In particular, CloudyFilms needs two large VMs to run the service portal and the streaming server, and a small VM to run the customer registry. It also needs separate storage spaces for the customer data and the video files (see Figure 2). Though there are many requirements that would be relevant for these resources, we have chosen a small subset of security requirements that are of particular interest for cloud computing and brokering:

1. *Data retention:* Data shall only be stored for the period required by the purpose for which they were collected. When the contract between the parties expires (or after a specific time), all consumer data must be deleted so that it cannot be recovered.
2. *Data location:* Data shall only be stored or processed in a location under the influence of the European privacy directive.
3. *Non-delegation:* The service provider shall not subcontract processing or storage to any third party.

These requirements are targeted towards specific resources. For example, the data retention requirement is imposed on the customer data storage (S1) to guarantee that customer data is completely removed from a provider's storage in case it is moved to a new provider. Moreover, the location requirement is imposed on both the customer data storage (S1) and the customer registry's VM (VM2) to ensure they will always be hosted in European datacentres. Finally, the non-delegation requirement is imposed on the streaming server (VM3) and the video storage (S2). This requirement ensures that, if S2 and VM3 are replicated on several providers, the selected providers do not use the same infrastructure (i.e., preventing one of them from sub-contracting IaaS services from the other one, the latter becoming a single point of failure).

In this use case, we assume that there is not a single IaaS provider able to offer all the resources needed by CloudyFilms at an affordable price while at the same time fulfilling all the security requirements. Since dealing with multiple providers is too much of an administrative and technical burden for a small company, CloudyFilms decides to use the services of a cloud broker. The broker gets CloudyFilms' requirements on hardware, security and QoS and uses them to select the cheapest IaaS providers that fulfil those requirements (e.g., it selects provider 2, in Europe, to host the customer data, even though it is more expensive than provider 3, in the USA – see Figure 2). With each of the selected providers, the broker automatically negotiates an SLA on behalf of CloudyFilms and supervises its enforcement.

Figure 2 A cloud broker negotiating security requirements for CloudyFilms



4 Specifying security requirements in SLAs

In this section, we review a selection of ten existing deontic contract languages found in the literature and for each of them we try to express at least one of the sample security requirements from the use case in Section 3. This is not intended to be an exhaustive survey, but a sample used to illustrate the diversity and suitability of current technology for expressing security requirements in a machine readable way in the context of service brokering for cloud SLAs. It is worth noting that many of the reviewed languages are not directly comparable, since they are intended for somewhat different purposes. For instance, some of the languages are intended for discovery and negotiation, others formalise already agreed-upon contracts, and there are languages used for evaluating service performance and triggering appropriate responses in cases where there are violations. Some of them act as versatile containers to be used in combination with more domain specific languages, while others have been created without much alignment towards other languages and standards. In addition, there are all sorts of overlap in this landscape.

4.1 SLAng

SLAng originated from the EU project Tapas (Lamanna et al., 2003a), which ended in 2005, and was an early initiative to formalise SLAs. It was developed to capture mutual responsibilities, specifically QoS properties, between service providers and clients. The language consists of three general components, namely a *service description* (e.g., service location and provider information), *contract statements* (e.g., duration, penalties) and *service level specifications* (i.e., technical QoS descriptions and metrics). It has a well-defined XML syntax and semantics, and can express different levels of non-functional features (e.g., application, resources) between independent parties. This allows for tier-specific horizontal and vertical SLAs between service consumers and providers. The format of the language was created to support negotiation, contract agreements and automatic reasoning/adaptation (Lamanna et al., 2003b). In Listing 1 we have expressed that the provider of storage for customer data (S1) must be located in Europe and that the data must also be located in Europe. Data retention is *no-retention* (what this means must be defined elsewhere), but delegation is explicitly allowed.

SLAng is a well-known language in academia, giving inspiration to related work, but has neither been subject to industrial uptake nor standardisation. Though the Web page of the Tapas project does not exist any more, the public reports can still be accessed through Sourceforge (see http://tapas.sourceforge.net/). We consider the language itself fairly easy to use and understand, but as pointed out by Lamanna et al. (2003b), further work is necessary on the definition of the semantics of SLAng. In most cases the language is too abstract to specify an SLA, therefore it must be extended with more domain specific information (Skene et al., 2010). This is exactly what we had to do in Listing 1, since the XML attributes *data_location*,*data_retention* and *delegation* are not part of the formal semantics of SLAng. The language has been created to support security properties, but these are mostly of a measurable and quantifiable nature like *availability* and *encryption type*. The language needs to be contained in a wider contract language since it is domain specific for QoS properties.

Listing 1 Expressing data retention and delegation requirements using SLAng

```
<?xml version=‘‘1.0” encoding=‘‘UTF−8”?>
<SLAng xmlns:xsi=‘‘http://www.w3.org/2001/XMLSchema−instance”
     xsi:noNamespaceSchemaLocation=‘‘dave/TAPAS/SLAng0_4/SLAng0_4.xsd”>
  <Vertical>
    <Application>
      <Id sls_id=‘‘123” service_id =‘‘S1”/>
      <Client>
        <Name>CloudyFilms</Name>
      </Client>
      <Server>
        <Name>S1 provider</Name>
        <Place>Europe</Place>
        <Security data_location =‘‘Europe” data_retention =‘‘no−retention”
             delegation =‘‘anyone”/>
      </Server>
    </Application>
  </Vertical>
</SLAng>
```

4.2 P3P (APPEL)

The platform for privacy preferences (P3P) specification (Cranor, 2003) was created to allow websites to express their privacy policies in machine readable format. The purpose is to inform the user about how the personal information that is collected by the website will be handled. Privacy preferences are expressed using APPEL (Cranor et al., 2002), which are evaluated against the P3P policy files in order to let user agents make automated or semi-automated decisions.

For the CloudyFilms use case, the APPEL Language can be used to place restrictions on how the customer data stored by the provider will be used, in terms of *purpose*, *recipient* and *retention*. The rule set in Listing 2 gives a data retention requirement stating that it is not acceptable that the service provider keeps purchase information indefinitely. Here we have assumed that the customer data consists of physical and online contact information, as well as purchase information.

Listing 2 An APPEL rule set for restricting access to customer data

```
<appel:RULE behavior=‘‘block” description =‘‘Service may keep
     customer data indefinitely ”>
 <POLICY>
 <STATEMENT appel:connective=‘‘and”>
  <DATA−GROUP><DATA>
   <CATEGORIES appel:connective=‘‘or”>
    <physical/><online/><other−category/>
   <CATEGORIES>
  </DATA></DATA−GROUP>
  <RETENTION><indefinitely/></RETENTION>
 </STATEMENT>
</POLICY>
</appel:RULE>
```

It is straightforward to place restrictions on data storage retention using APPEL, however, it is not possible to articulate restrictions on the geographic location. APPEL also has limited support for expressing the non-delegation requirement. Writing policies in P3P/APPEL is considered fairly easy and there are several tools that help translate policies stated in natural languages to the machine-readable format. The language is easily extensible, however, the restricted vocabulary and the limited scope (data collection practices for websites) makes it difficult to express security requirements for service-oriented architectures such as the cloud.

Version 1.0 of the P3P specification was released in 2002, representing a cornerstone in the online privacy protection field. However, adoption was slow from the beginning, and a study performed four years after the introduction showed that only a fraction of all websites had a P3P policy (Egelman et al., 2006). The work on the specification has been officially suspended since 2006.

4.3 RBSLA

The Rule-based Service Level Agreement Language (RBSLA) (Paschke, 2005) is a declarative mark-up language for rule-based policy and contract specifications based on

RuleML (2012) and logic programming. Its intended application is to formalise contracts so that analysis and monitoring of contract performance can be carried out at runtime. Things like rule chaining and behaviour in cases of failure can be expressed as a part of the contract logic. It was designed to be compatible with the Semantic Web and other existing standards, but has not become a standard itself. The initiative seems to have lost its momentum some years ago, since there are no recent publications, and there have been no updates to the supporting management tools (located on Sourceforge) and the RBSLA web page since 2006. The language itself has high expressiveness, but as with other languages based on logic programming, it might be difficult to understand and use by non-experts. A higher level representation that can be transformed into RBSLA is therefore needed in most cases. Given the limited available documentation for RBSLA we were not able to create an example listing for any of the use case study requirements.

4.4 Common policy (RFC 4745)

Common policy is better known as RFC 4745 (Schulzrinne et al., 2007), and is a framework for creating authorisation policies for access to application-specific data. The purpose of the framework is to combine location specific policies and presence specific policies into one common authorisation system. The framework defines a rule set (i.e., policy), consisting of *conditions*, *actions* and *transformations* (i.e., permissions), that are evaluated in order to determine if a request for access to data items should be permitted or not. There are currently three conditions defined in RFC4745: *identity*, *sphere* and *validity*; making it possible to put restrictions on *who, where* and *when* data can be accessed. RFC4745 does not directly support any of the example security requirements for the CloudyFilms use case, but the identity condition may be used to put restrictions on the domain attribute for providers who try to access the data. For example, the rule set in Listing 3 will give access to any (authenticated) provider, except to service providers within the domains example.com and example.org.

Listing 3 A rule set expressed according to RFC 4745

```
<rule id=‘‘f3g44r1”>
  <conditions>
    < identity >
      <many>
        <except domain=‘‘example.com”/>
        <except domain=‘‘example.org”/>
      </many>
    </ identity >
 </conditions>
<actions/>
<transformations/>
</rule>
```

The purpose of RFC4745 was to increase interoperability by allowing authorisation policies to travel with the data, and could possibly be extended to be applicable to the cloud brokering context. The framework has already been extended and implemented as part of presence-based systems based on SIP (IBM, 2009).

4.5 XACML

eXtensible Access Control Markup Language (XACML) (OASIS, 2005) describes both a policy language and an access control decision language. It is normally used to grant permission in order to perform an action on a resource. XACML can also be used to find a policy that applies to a given request and evaluate the request against the policy.

The current 2.0 version of XACML is very limited for other purposes than access control. However, in WS-XACML, which is a proposed feature for XACML 3.0, the client (i.e., customer) can use an XACMLPrivacyAssertion to make sure that the service fulfils an obligation regarding the client's provided personal information. Two such assertions will match if every requirement in each assertion is satisfied by at least one capability in the other. The requirement in Listing 4 (expressed in pseudo code) states that customer data should not be stored by the provider or forwarded to any third party.

Listing 4 A rule set expressed in WS-XACML

```
XACMLPrivacyAssertion
 Requirements
  RETENTION: data kept only until transaction completed
  RECIPIENT: data not given to any 3rd party
 Capabilities
  Provide customer data
```

XACML is very powerful and is easily extensible, but writing and reasoning over XACML policies is considered difficult. It has been widely adopted, especially within academic research. Version 3.0 will include additional aspects, such as delegation, which will make it possible to delegate access policies and put constraints of the delegation.

4.6 WS-Agreement

The WS-Agreement specification (Andrieux et al., 2003) is a protocol for establishing an agreement between two parties, such as service providers and consumers. It allows the use of any service term, and is therefore suitable for security agreements as well. The specification provides a template for the agreement, which consists of the name of the agreement (this is optional), the context (i.e., the participants and the lifetime of the agreement) and the agreement terms. The agreement terms are used to specify the obligations of the parties and the associated guarantee terms are used to provide assurance to the service consumer of the service quality and/or resource availability offered by the service provider.

WS-Agreement describes three main elements: the name/ID, the context element and the terms section. The terms section is the core part of the agreement and is used to specify the obligations of the parties. The terms section consists of two different parts: the service terms, which are used to describe the functional aspects of the service (i.e., its interface description and its endpoint reference); and the guarantee terms, which capture the monitorable aspects of the service that can fail independently of the functioning of the core service (Ludwig et al., 2006). A guarantee term has the following elements:

- *the service scope*, which is used to define what servicethe terms apply to
- *the qualifying condition*, which defines under what conditions the guarantee applies
- *the service level objective*, which defines the actual guarantee
- *the business value list*, which is used to express different value aspects of the service level objectives, which in a cross-organisational setting usually are stated as penalty or reward values.

WS-Agreement does not include any ontology for expressing security requirements, but it is possible to use the service level objectives in the guarantee terms to put restrictions on data storage location, retention and non-delegation using any existing security ontologies. For example, a service level objective that uses P3P to put restrictions on the data retention could look as illustrated in Listing 5. Note that to save space we did not include the qualifying condition or the business value list in this example.

Listing 5 A WS-Agreement service level objective that uses P3P to put restrictions on data retention

```
<wsag:GuaranteeTerm
 wsag:Name=''SecurityRequirements''
 wsag:Obligated='' Provider''>
 <wsag:ServiceScope
  ServiceName=''StoreConsumerData''>
 </wsag:ServiceScope>*
 <wsag:ServiceLevelObjective>
  <wsag:CustomServiceLevel>
    <RETENTION><stated-purpose/></RETENTION>
  </wsag:CustomServiceLevel>
 </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

WS-Agreement is an open standard and it has been widely adopted for QoS support for service-oriented architectures in web and grid contexts.

4.7 PrimeLife Policy Language

The PrimeLife Policy Language (PPL) is an XML-based policy language developed in the EU project PrimeLife (PrimeLife Consortium, 2012). In addition to access control, PPL provides data handling as an extension to XACML 3.0 (Ragget et al., 2009). PPL is focused around data handling and credential capabilities. The user's privacy preferences are evaluated against the service provider's data handing policies; if there is a match, a sticky policy can be attached to the data. PPL supports both data retention and non-delegation; the latter by making it possible for a user to specify to whom and under what circumstances personal data may be forwarded to a third party (called 'downstream data controller' in the language specification). Non-delegation using PPL is expressed in Listing 6, where the data handling policy AuthorizationsSet is used to define what the service provider can do with the collected information and the ObligationsSet defines the obligations that the provider promises to honour. The current draft of PPL does not support restrictions in terms of geographic location of the data storage and processing, but the vocabulary is left open and might be extended to include such restrictions. PPL is still very young since PrimeLife has just completed, and there is consequently little adoption.

Listing 6 Expressing non-delegation in PPL

```
<DataHandlingPreferences>
 <ObligationsSet> .. </ObligationsSet>
 <AuthorizationsSet>
   ..
 <AuthzDownstreamUsage allowed=‘‘false”>
 </AuthzDownstreamUsage>
 </AuthorizationsSet>
```

4.8 Business Contract Language

The Business Contract Language (BCL) (Governatori and Milosevic, 2006) is an event driven language intended for runtime monitoring of contract terms. A single event can be used to signify actions of the signatories, temporal occurrences or change of state associated to a contract variable. Listing 7 shows how one may express the data retention requirement as an *obligation* (i.e., pattern that must occur given an event) for the provider.

Listing 7 A data retention requirement expressed in BCL

```
Policy: DataRetention
  Role: Provider
  Modality: Obligation
  Trigger: StoreConsumerData
  Behaviour: DeleteConsumerData after StoreConsumerData.date + 21
```

Besides obligations, modality can be used to express *permissions* (i.e., allowed behaviour) and *prohibitions* (i.e., what must not occur). Given this expressiveness, all three use case requirements should be possible to represent in BCL. The language also supports the expression of violations and their corresponding reparations, which is very relevant for security policies. The language therefore seems well suited for security SLA negotiation, and still manages to have a limited set of constructs. It seems fairly easy to extend, but there must be a common agreement on triggers and behaviour among the involved parties. Though BCL was introduced in 2005, there is currently little available information, tools and activities related to it.

4.9 ConSpec

The ConSpec Language (Aktug and Naliuka, 2008; Greci et al., 2009) can be used to formally specify contracts and various security enforcement tasks, and it is strongly inspired by the policy specification language PSLang (Erlingsson, 2004) for runtime monitoring. It consists of declarations related to the security state and, like BCL, defines events that trigger actions for updating the states. Listing 8 exemplifies the non-delegation requirement. No specific variables are defined after the security state declaration, but the event clause tells us that the service can only be invoked as long as it promises to not delegate the task anyone else.

Listing 8 A non-delegation requirement in ConSpec

```
SCOPE composition
SECURITY STATE
BEFORE invokeService(s)
PERFORM
  (s.delegation().equals(''none'')) -> skip
```

ConSpec is a relatively simple and restricted language intended for expressing security requirements, with a finite set of variables, and no loops. It is well suited for contract matching, can be somewhat extended, but is to this date mostly limited to academic use. However, parsing tools are available, and adoption and documentation is currently being done through the Aniketos project (Aniketos Consortium, 2012).

4.10 eContracts

LegalXML eContracts (Leff and Meyer, 2007) is an OASIS open standard for the markup of contract documents to enable creation, maintenance, management, exchange and publication of contract documents and contract terms. It is intended to be used by automated processing systems rather than lawyers, and to structure any kind of contract. A simplified example representing the location requirement using eContracts is shown in Listing 9. The grammatical content is represented inside the 'block' element, and the character data within the 'text' element. In this example, we have used a conditional attribute to express that it has a jurisdiction limited to the EU.

Listing 9 An eContracts specification with a location requirement

```
<?xml version=''1.0'' encoding=''utf-8''?>
<contract xmlns=''urn:oasis:names:tc:eContracts:1:0''>
  <title><text>Persistent storage location</text></title>
  <conditions><condition name=''EU''>European Union
        </condition></conditions>
  <body>
   <block condition=''EU''>
    <text>Data shall only be stored on servers located
         within the European Union</text>
   </block>
  </body>
</contract>
```

Though eContracts is a deontic contract language, the contractual terms within the blocks are still mostly defined using natural language. The reason is that eContracts is intended to represent contracts that have already been agreed upon and signed by the involved parties, abstracting from the specific word processing tool used. This makes eContracts less suitable for cloud SLA brokering. The specification defines 51 core elements, and provides a generic structure that can be used to encompass a wide range of contracts. The structure is simple, with high degree of freedom. Version 1.0 of the specification dates back to 2007, but at the same time the technical committee was dissolved, and to the best of our knowledge there has been little subsequent activity to continue the work of eContracts.

5 Comparison and classification of languages

In the previous section we reviewed ten different specification languages and investigated to what degree they are suitable to express the security requirements for the CloudyFilms use case. Table 1 summarises the characteristics and intended use of these languages, while Table 2 shows a comparison of the languages based on the following properties:

- *feasibility:* how well the language fits the type of requirements considered in the case study
- *complexity:* advanced features and required expertise needed to make use of the language (a simple language is considered to be more user friendly)
- *extensibility:* the possibility of adding additional concepts and expressions
- *maturity:* how long the language has been available and current stage in life
- *support:* associated documents, tools and other sources of information
- *adoption:* the current uptake among the relevant stakeholders.

As can be seen from Tables 1 and 2, all languages have their strengths and weaknesses. P3P, PPL, XACML and RFC4745 are all declarative domain specific languages, the former two intended for specifying data handling policies (i.e., privacy policies) and the latter two for specifying access control policies. P3P represents early work in the context of privacy protection and usage of personal information, and several of the subsequently developed languages have been inspired by this work. However, this initiative is more or less dead, mostly because the slow adoption and limited interest from the service providers. In the context of contractual agreements the main disadvantage of P3P is its limited application area (i.e., data handling policies and preferences), which is shared by PPL. XACML is designed to be more general-purpose than both P3P and PPL, which is both a strength and a weakness. Neither of the four domain specific languages are however intended to be used to construct SLAs and lack fundamental properties such as support for SLA negotiation, creation, assessment (monitoring) and violation handling. As illustrated in Table 3, a common drawback is that they cannot be used to put restrictions on data location.

On the contrary, WS-Agreement, BCL and ConSpec all seem suitable for expressing security requirements in a deontic form. These three languages support several of the phases in the SLA lifecycle discussed in Section 2. However, none of them specify a security term ontology, leaving the problem of translating security requirements stated in natural language to a machine-readable format unsolved.

Table 1 Characteristics and intended use of the languages in Section 4

	SLAng	*P3P*	*RFC*	*RBSLA*	*XACML*	*WS-Agr*	*PPL*	*BCL*	*ConSp.*	*eContr.*
Type of language										
Container						x				x
Declarative	x	x	x	x	x	x	x	x		
Logical									x	
Informal										x
Event-driven				x				x	x	
Intended use										
Service discovery	x					x			x	
SLA negotiation	x					x				
SLA creation	x					x			x	x
SLA assessment				x	x		x	x		
SLA viol. handl.				x				x	x	
Access control			x		x					
Privacy policy		x					x			

Table 2 Comparing the specification languages in Section 4

Language	*Feasibility*	*Complexity*	*Extensibility*	*Maturity*	*Support*	*Adoption*
SLAng	High	Medium	High	High	Low	Low
P3P	Medium	Low	Medium	High	Medium	Medium
RFC4745	Low	Low	High	Low	Low	Medium
RBSLA	Medium	High	High	High	Low	Low
XACML	Low	Medium	High	High	High	High
WS-Agr.	High	Low	High	High	High	High
PPL	Medium	Medium	High	Low	Low	Low
BCL	High	Medium	Medium	Medium	Low	Low
ConSpec	High	Medium	Medium	Medium	Low	Medium
eContracts	Low	Low	High	High	Low	Low

Table 3 Expressing the three security requirements with P3P, RFC4745, XACML and PPL

Language	*Data retention*	*Data location*	*Non-delegation*
P3P	yes	no	no
RFC4745	no	no	no
XACML	yes	no	yes
PPL	yes	no	yes

RBSLA, BCL and ConSpec are event-driven languages, which means that they are suitable for the monitoring and violation handling parts of the SLA lifecycle management. However, they all lack support for the fundamental parts of cloud SLA management. RBSLA and BCL do not support service discovery, SLA negotiation and the creations of contracts, while ConSpec does not support SLA negotiation and lacks support for other non-functional attributes than security.

WS-Agreement and eContracts are both container languages, which means that they lay out the structure and types of content in the different parts of the contract but without specifying the details of the actual terms in the contract. While WS-Agreement is intended to be used by automatic processing systems, and therefore supports creating agreement based on offers, an eContracts document is intended to represent an already signed contract.

6 Related work

There are several other efforts on contract languages and SLA frameworks which we briefly outline below.

6.1 More on contract languages

This article has investigated ten different deontic contract languages. However, there are numerous other languages that we have not explored, including for instance ecXML (Farrell et al., 2004), which has an event-based nature similar to BCL. The Contract Expression Language (Wang, 2010) is similar to eContracts, meaning that it is designed to express already agreed upon terms between the involved parties. Web service level agreements (WSLA) was designed to be a flexible SLA definition language, but is not suitable for security due to its focus on downtime, throughput, response time and other quantifiable parameters (WSLA, 2003). Somewhat related to our brokering use case, Nepal et al. (2009) present an XML-based contract language for establishing collaborative services. Although their approach seems more geared towards an environment of collaborating peers, they do describe a situation where collaborators contribute through providing web services. They provide little details on security aspects, but highlight the need for deletion of information after a contract termination. A wider range of related policy languages and protocols can be found in papers by Yagüe (2006) and Dwivedi and Padmanabhuni (2008). Jureta et al. (2009) mention a plethora of approaches for describing service quality, such as quality-value-dependency-priority model (QVDP), Quality Markup Language (QML), Web-services Offering Language (WSOL), WSLA, quality of service-aware component architecture (QuA), uniframe, Corba object trading service and quality objects (QuO).

According to Pearson and Charlesworth (2009), translation of legislation/regulation to machine readable policies has proven to be very difficult, and they give an overview of various projects that have tried to do so, including privacy incorporated software agent (PISA), Sparcle, REALM, LegalXML, and their own Encore project (Encore Consortium, 2008).

6.2 SLA frameworks for cloud computing

A number of frameworks for automatic SLA management have been proposed, which differ in their application domain, the type of negotiation protocols supported (i.e., single-round, multi-round or auction-like) and their architecture (i.e., whether they propose a centralised broker, a distributed marketplace, etc.). The state of the art in broker-based SLA management includes the work done in four European research projects during the last decade, which we review below.

The SLA@SOI project (SLA@SOI Consortium, 2009) ran from 2009 to 2011 and envisioned an open, dynamic, SLA-aware market for internet service providers. To achieve that vision, the project developed an open framework for management of

SLAs through the entire service lifecycle. The framework supports multi-layered SLA management, where SLAs can be composed and decomposed along functional and organisational domains (Happe et al., 2011). It also supports different SLA negotiation protocols and can be easily adapted to different application domains. The latter is possible thanks to a generic and language-agnostic SLA model (Kearney and Torelli, 2011) that generalises and refines some of the concepts in the WS-Agreement, WSLA and WSDL standards. This model leaves the specification of QoS service terms open, although a limited set of standard QoS terms (e.g., for availability) have been defined. Specifically related to this model is their SLA-enabled reference architecture, which included requirements on compliance, transparency, security, privacy, auditability and data encryption.

The BREIN project (BREIN Consortium, 2006) developed an intelligent grid infrastructure enabling companies to collaborate in a dynamic e-business environment. SLA management, based on so-called semantically annotated SLAs (SA-SLAs) (Koller et al., 2010), plays an important role in the proposed infrastructure. SA-SLAs enable SLA-based discovery and SLA negotiation in open markets, where different parties use different terminologies. They are based on a combination of both WS-Agreement, used as a container, and WSLA, used to describe the domain-specific service and guarantee terms, which are semantically annotated.

The OPTIMIS project (OPTIMIS Consortium, 2010) aims at defining an architectural hybrid cloud framework and toolkit enabling companies to easily move services and applications from private clouds to trustworthy and auditable public clouds. The envisioned toolkit will allow the implementation of different cloud architectures, including a broker-based architecture as the one in the paper by Ferrer et al. (2012). OPTIMIS has adopted WS-Agreement and WS-Agreement negotiation (Wäldrich et al., 2011) for SLA specification and negotiation, and is defining its own XML schema to describe service terms for trust, risk, eco-efficiency and cost parameters. The OVF specification is used as term language to express data security and data centre placement requirements (Ziegler and Jiang, 2011).

The mOSAIC project (mOSAIC Consortium, 2012) intends to create an open-source cloud API and platform targeted for developing SLA-aware multi-cloud oriented applications. From the end-user's point of view, the main component is the cloud agency, a broker that will assist applications in discovering cloud resource providers, negotiating SLAs with these providers, and monitoring the SLA fulfilment. The platform will use an ontology for cloud services (Moscato et al., 2011), where security is included as a non-functional requirement, but only on a high level. mOSAIC has chosen WS-Agreement to describe SLAs and SLA templates, and OCCI (Open Grid Forum, 2009) as a domain specific language to describe requirements on cloud resources.

7 Towards cloud SLAs with security requirements

From our investigation of the ten different deontic contract languages one can conclude that there is no silver bullet when it comes to specifying security requirements in SLAs. We are not aware of any existing language that can be used in its current form to create a machine-readable contract that guarantees the fulfilment of our requirements on data location, retention and non-delegation for the CloudyFilms use case. Additionally, the different languages represent different levels of abstraction. This is important, since in practice, a cloud customer will most likely want to include other non-functional requirements as well in the agreement, for example the availability, performance and service cost.

We consider a combination of a container language with a domain specific language to be the best solution for expressing security requirements for cloud services in a machine-readable format. Out of the existing container languages that were evaluated in Sections 4 and 5, we consider WS-Agreement to be the most promising candidate. There are several reasons for this:

- WS-Agreement is well-known, widely accepted and is frequently being used, mostly by the research communities but also in some commercial projects. Several implementations exist. In addition, WS-Agreement is an open standard.
- WS-Agreement is easily extensible. The templates provided in the standard include the main concepts of an agreement and the necessary language elements that describe the main elements at the top level. The details of the agreement can then be filled with other domain specific languages. This makes it possible to add any kind of non-functional requirement to the agreement.
- WS-Agreement is more than just a contract language. It is compliant with all steps in the SLA management lifecycle, including support for service discovery, SLA negotiation and establishment, monitoring of the service terms and termination.

The WS-Agreement specification does not define which domain specific language (DSL) to use to express the service level objectives in an agreement. This must eventually be agreed upon. There is also a need for an ontology that captures the concepts used in security requirements. In this way we can represent the required and offered security properties for a service in an unambiguous and language independent way. We have examined several potential candidates for a security requirements DSL in this article. However, as indicated in Table 3, none of them can be used to express the three security requirements (i.e., data retention, data location and non-delegation) that were considered necessary for CloudyFilms, which was a pretty simple case study. As far as we are aware, there is no existing DSL that has been designed to express security properties of cloud services. To remedy this, we have initiated work on an XML schema to describe the structure of the data retention, data location and non-delegation security requirements. In Listing 10 we outline a first draft of the schema. A complete DSL for cloud security SLAs must, however, include all potential security controls that cloud providers and customers would like to see included. An initial framework for cloud security SLAs has been presented by Bernsmed et al. (2011). This framework is intended to be put in concrete form using existing security control frameworks, such as the ENISA cloud computing assurance framework (ENISA, 2009b), NIST SP 800-144 (Jansen and Grance, 2011) and ISO27002 (ISO/IEC 27002, 2005).

Listing 10 A possible schema definition for the three security requirements

```
<xs:simpleType name=‘‘DataRetention”>
<xs:annotation>
<xs:documentation>
The DataRetention condition specifies how long data will be retained by
       the service provider: no−retention, stated −purpose (data discarded
       at the earliest time possible); legal −requirement (as required
       by law or liability under applicable law); business−practice
       ( retention according to service provider’s business practices ,
       with an explicit destruction time table); and indefinitely .
</xs:documentation>
</xs: annotation >
      <xs: restriction base=‘‘xs: string ”>
            <xs:enumeration value=‘‘no−retention”/>
            <xs:enumeration value=‘‘ stated −purpose”/>
            <xs:enumeration value=‘‘ legal −requirement”/>
            <xs:enumeration value=‘‘business practices ”/>
            <xs:enumeration value=‘‘ indefinitely ”/>
      </xs: restriction >
</xs:simpleType>

<xs:simpleType name=‘‘DataLocation”>
<xs:annotation>
<xs:documentation>
The DataLocation condition specifies whether transfer and storage of the
       data must be restricted to countries that are members of the EU (‘EU’),
       to countries that may not be EU member states but that do implement the
       data protection directive (‘DPD’) or if there are no such restrictions (‘None’).
</xs:documentation>
</xs: annotation >
      <xs: restriction base=‘‘xs: string ”>
            <xs:enumeration value=‘‘EU”/>
            <xs:enumeration value=‘‘DPD”/>
            <xs:enumeration value=‘‘None”/>
      </xs: restriction >
</xs:simpleType>

<xs:simpleType name=‘‘Delegation”>
<xs:annotation>
<xs:documentation>
The Delegation condition specifies whether the service provider is allowed
       to delegate the service to any other provider (‘anyone’),
       to providers accountable to the original provider that follow different
       security practices (‘ others ’), to providers accountable to the original
       provider that follow equal security practices (‘same’), or if it is not
       allowed to delegate the service at all (‘none’).
</xs:documentation>
</xs: annotation >
      <xs: restriction base=‘‘xs: string ”>
            <xs:enumeration value=‘‘none”/>
            <xs:enumeration value=‘‘same”/>
            <xs:enumeration value=‘‘ others ”/>
            <xs:enumeration value=‘‘anyone”/>
      </xs: restriction >
</xs:simpleType>
```

8 Conclusions

Current cloud SLAs are written in a natural language form, and seldom cover security requirements. This is hindering the uptake of cloud computing, since establishing appropriate contracts satisfying security policies is considered to be difficult and time consuming. We envision cloud brokers to have an important role in helping customers to find the most suitable cloud providers in a more automated and dynamic way. The brokers would translate customer requirements into deontic contract languages, and automatically negotiate and monitor SLAs with cloud providers on behalf of the customers. Many different contract specification languages exist today. We have reviewed ten of these and found that there is no single 'silver bullet' language that stands out as a prevalent candidate. Further work is required on the specification and reasoning of security requirements for cloud SLA brokering, and there is a need for a common ontology that represents contractual security concepts.

Acknowledgements

The research leading to these results has been supported by Telenor through the SINTEF-Telenor research agreement and the European Union Seventh Framework Programme (FP7/2007-2013) under Grant No. 257930.

References

Aktug, I. and Naliuka, K. (2008) 'ConSpec – a formal language for policy specification', *Electron. Notes Theor. Comput. Sci.*, Vol. 197, No. 1, pp.45–58.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S. and Xu, M. (2003) 'Web services agreement specification (WS-Agreement)' [online] https://forge.gridforum.org/projects/graap-wg/ (accessed 12 October 2012).

Aniketos Consortium (2012) 'Aniketos – secure and trustworthy composite services' [online] http://www.aniketos.eu/ (accessed 12 October 2012).

Bernsmed, K., Jaatun, M.G. and Undheim, A. (2011) 'Security in service level agreements for cloud computing', in *Proc. 1st International Conference on Cloud Computing and Services Science, (CLOSER 2011).*

BREIN Consortium (2006) 'BREIN FP6 EU Project' [online] http://www.eu-brein.com/ (accessed 12 October 2012).

Cranor, L.F. (2003) 'P3P: making privacy policies more useful', *Security & Privacy, IEEE*, Vol. 1, No. 6, pp.50–55.

Cranor, L.F., Langheinrich, M. and Marchiori, M. (2002) 'A P3P Preference Exchange Language 1.0 (APPEL1.0)', World Wide Web Consortium [online] http://www.w3.org/TR/P3P-preferences/ (accessed 12 October 2012).

Dwivedi, V. and Padmanabhuni, S. (2008) 'Providing web services security sla guarantees: issues and approaches', in Khan, K.M. (Ed.): *Managing Web Service Quality: Measuring Outcomes and Effectiveness*, Chapter 13, pp.286–305, IGI Global.

Egelman, S., Cranor, L.F. and Chowdhury, A. (2006) 'An analysis of P3P-enabled web sites among top-20 search results', in *Proceedings of the 8th Int. Conf. on Electronic Commerce*, ICEC '06, pp.197–207.

Elmroth, E., Marquez, F.G., Henriksson, D. and Ferrera, D.P. (2009) 'Accounting and billing for federated cloud infrastructures', in *Proceedings of the 2009 Eighth International Conference on Grid and Cooperative Computing*, GCC '09, pp.268–275.

Encore Consortium (2008) 'Encore project' [online] http://www.encore-project.info/ (accessed 12 October 2012).

ENISA (2009a) 'Cloud computing: benefits, risks and recommendations for information security' [online] http://www.enisa.europa.eu/activities/risk-management/files/deliverables/ cloud-computing-risk-assessment (accessed 12 October 2012).

ENISA (2009b) 'Cloud computing information assurance framework' [online] http://www.enisa.europa.eu/activities/risk-management/files/deliverables/ cloud-computing-information-assurance-framework/ (accessed 12 October 2012).

Erlingsson, U. (2004) *The Inlined Reference Monitor Approach to Security Policy Enforcement*. PhD thesis, Ithaca, NY, USA. AAI3114521.

Farrell, A. D.H., Sergot, M.J., Trastour, D. and Christodoulou, A. (2004) 'Performance monitoring of service-level agreements for utility computing using the event calculus', in *Proc. First IEEE Int. WS on Electronic Contracting*, pp.17–24.

Ferrer, A.J., Hernández, F., Tordsson, J., Elmroth, E., Ali-Eldin, A., Zsigri, C., Sirvent, R., Guitart, J., Badia, R.M., Djemame, K., Ziegler, W., Dimitrakos, T., Nair, S.K., Kousiouris, G., Konstanteli, K., Varvarigou, T., Hudzia, B., Kipp, A., Wesner, S., Corrales, M., Forgó, N., Sharif, T. and Sheridan, C. (2012) 'Optimis: a holistic approach to cloud service provisioning', *Future Generation Computer Systems*, Vol. 28, No. 1, pp.66–77.

Finnegan, J., Malone, P., Maranon, A. and Guillen, P. (2007) 'Contract modelling for digital business ecosystems', in *Digital EcoSystems and Technologies Conference, DEST '07*, pp.71–76.

Gartner (2011) 'Public cloud services, worldwide and regions, industry sectors, 2010–2015', 2011 Update [online] http://softwarestrategiesblog.com/2011/07/02/sizing-the-public-cloud-services-market/ (accessed 12 October 2012).

Governatori, G. and Milosevic, Z. (2006) 'A formal analysis of a business contract language', *Int. J. Cooperative Inf. Syst.*, Vol. 15, No. 4, pp.659–685.

Greci, P., Martinelli, F. and Matteucci, I. (2009) 'A framework for contract-policy matching based on symbolic simulations for securing mobile device application', in *Leveraging Applications of Formal Methods, Verification and Validation*, Vol. 17 of *Communications in Computer and Information Science*, pp.221–236, Springer, 10.1007/978-3-540-88479-8_16.

Happe, J., Theilmann, W., Edmonds, A. and Kearney, K.T. (2011) 'A reference architecture for multi-level SLA management', in Wieder, P., Butler, J.M., Theilmann, W. and Yahyapour, R. (Eds.): *Service Level Agreements for Cloud Computing*, pp.13–26, Springer, New York.

Heiser, J. and Cearley, D.W. (2011) 'Hype cycle for cloud security', Research Note G00214151.

IBM (2009) 'General considerations for setting up security for presence server' [online] http://publib.boulder.ibm.com/infocenter/wtelecom/v7r0m0/index.jsp?topic=/ com.ibm.presence.plan.doc/generalsecurity_c.html (accessed 12 October 2012).

ISO/IEC 27002 (2005) 'Information technology – security techniques – code of practice for information security management' [online] http://www.iso.org/iso/catalogue_detail?csnumber=50297 (accessed 12 October 2012).

Jansen, W. and Grance, T. (2011) 'Guidelines on security and privacy in public cloud computing' [online] http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf (accessed 12 October 2012).

Jureta, I.J., Herssens, C. and Faulkner, S. (2009) 'A comprehensive quality model for service-oriented systems', *Software Quality Control*, Vol. 17, No. 1, pp.65–98.

Kearney, K.T. and Torelli, F. (2011) 'The SLA model', in Wieder, P., Butler, J.M., Theilmann, W. and Yahyapour, R. (Eds.): *Service Level Agreements for Cloud Computing*, pp.43–67. Springer, New York.

Koller, B., Frutos, H.M. and Laria, G. (2010) 'Service level agreements in BREIN', in Wieder, P., Yahyapour, R. and Ziegler, W. (Eds.): *Grids and Service-Oriented Architectures for Service Level Agreements*, pp.157–165, Springer, USA.

Lamanna, D.D., Skene, J. and Emmerich, W. (2003a) 'D2: specification language for service level agreements', Technical report, IST Project 34069 Tapas [online] http://tapas.sourceforge.net/ (accessed 12 October 2012).

Lamanna, D.D., Skene, J. and Emmerich, W. (2003b) 'Slang: a language for defining service level agreements', in *Proceedings of the Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, FTDCS '03*, p.100, Washington, DC, USA, IEEE Computer Society.

Leff, L. and Meyer, P. (2007) 'eContracts Version 1.0', Technical report, OASIS [online] http://docs.oasis-open.org/legalxml-econtracts (accessed 12 October 2012).

Lheureux, B.J. and Plummer, D.C. (2011) 'Cool vendors in cloud service brokerage', April 27, ID Number: G00212316, 8pp. [online] http://www.gartner.com/id=1657015 (accessed 12 October 2012).

Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L. and Leaf, D. (2011) *NIST Cloud Computing Reference Architecture*, NIST Special Publication 500-292.

Ludwig, A., Braun, P., Kowalczyk, R. and Franczyk, B. (2006) 'A framework for automated negotiation of service level agreements in services grids', in Bussler, C. and Haller, A. (Eds.): *Business Process Management Workshops*, Vol. 3812 of *Lecture Notes in Computer Science*, pp.89–101, Springer, Berlin/Heidelberg, 10.1007/11678564_9.

mOSAIC Consortium (2012) 'mOSAIC Cloud' [online] http://www.mosaic-cloud.eu/ (accessed 12 October 2012).

Moscato, F., Aversa, R., Martino, B.D., Fortis, T-F. and Munteanu, V.I. (2011) 'An analysis of mosaic ontology for cloud resources annotation', in *Procs. of Federated Conference on Computer Science and Information Systems (FedCSIS '11)*, pp.973–980, IEEE CS.

Nepal, S., Zic, J. and Chen, S. (2009) 'A contract language for service-oriented dynamic collaborations', in *Collaborative Computing: Networking, Applications and Worksharing*, Vol. 10 of *LNICST*, pp.545–562, Springer, 10.1007/978-3-642-03354-4_41.

OASIS (2005) 'eXtensible Access Control Markup Language (XACML) Version 2.0', OASIS Open [online] http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf (accessed 12 October 2012).

Open Grid Forum (2009) 'Open cloud computing interface (OCCI)' [online] http://forge.ogf.org/sf/projects/occi-wg (accessed 12 October 2012).

OPTIMIS Consortium (2010) 'OPTIMIS FP7 EU Project' [online] http://www.optimis-project.eu/ (accessed 12 October 2012).

Paschke, A. (2005) 'RBSLA – a declarative rule-based service level agreement language based on RuleML', in *Int. Conf. Comp. Intelligence for Modelling, Control and Automation, and Intelligent Agents, Web Tech. and Internet Commerce*, Vol. 2, pp.308–314.

Pearson, S. and Charlesworth, A. (2009) 'Accountability as a way forward for privacy protection in the cloud', in *Proc. 1st International Conference on Cloud Computing, CloudCom '09*, Springer, pp.131–144.

PrimeLife Consortium (2012) 'Primelife – privacy and identity management in europe for life' [online] http://www.primelife.eu/ (accessed 12 October 2012).

Ragget, D., Ardagna, C., Bournez, C., Bussard, L., Bezzi, M., Camenisch, J., deCapitanidi Vimercati, S., Kuczerawy, A., Meissner, S., Neven, G., Paraboschi, S., Pedrini, E., Pinsdorf, U., Preiss, F-S., Trabelsi, S., Tziviskou, C., Raggett, D., Roessler, T., Samarati, P., Schallaboeck, J., Short, S., Sommer, D., Verdicchio, M. and Wenning, R. (2009) 'H5.3.2 – draft 2nd design for policy languages and protocols', Technical report, The PrimeLife project [online] http://primelife.ercim.eu/results/documents/120-h532-draft-2nd-design-for-policy-languages-and-protocols (accessed 12 October 2012).

Ried, S. (2011) 'Cloud broker – a new business model paradigm', Technical report, Forrester Research.

RuleML (2012) 'The rule markup initiative' [online] http://www.ruleml.org.

Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk, J. and Rosenberg, J. (2007) 'Common policy: a document format for expressing privacy preferences', Request For Comments 4745 [online] http://tools.ietf.org/html/rfc4745.

Skene, J., Raimondi, F. and Emmerich, W. (2010) 'Service-level agreements for electronic services', *IEEE Trans. Softw. Eng.*, Vol. 36, No. 2, pp.288–304.

SLA@SOI Consortium (2009) 'SLA@SOI FP7 EU project' [online] http://sla-at-soi.eu/.

TeleManagement Forum (2005) *SLA Management Handbook*, Vol. 2 – Concepts and Principles.

Tøndel, I.A., Nyre, Å.A. and Bernsmed, K. (2011) 'Learning privacy preferences', in *Proceedings of the 6th Conference on Availability, Reliability and Security (AReS)*.

Tordsson, J., Montero, R.S., Moreno-Vozmediano, R. and Llorente, I.M. (2012) 'Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers', *Future Generation Computer Systems*, Vol. 28, No. 2, pp.358–367.

Wäldrich, O., Battré, D., Brazier, F. M.T., Clark, K.P., Oey, M.A., Papaspyrou, A., Wieder, P. and Ziegler, W. (2011) 'WS-Agreement negotiation: Version 1.0', Technical report, Open Grid Forum, Grid Resource Allocation Agreement Protocol (GRAAP) WG.

Wang, X. (2010) 'Specifying the business collaboration framework in the contract expression language', *International Journal of Business Process Integration and Management*, Vol. 4, No. 3, pp.200–208.

WSLA (2003) 'Web service level agreements (WSLA) project' [online] http://www.research.ibm.com/wsla/.

Yagüe, M.I. (2006) 'Survey on XML-based policy languages for open environments', *Journal of Information Assurance and Security*, Vol. 1, No. 1, pp.11–20.

Ziegler, W. and Jiang, M. (2011) 'OPTIMIS SLA framework and term languages for SLAs in cloud environment', OPTIMIS Project Deliverable D2.2.2.1.