EXPRESSING CLOUD SECURITY REQUIREMENTS IN DEONTIC CONTRACT LANGUAGES

Per Håkon Meland[1], Karin Bernsmed[1], Martin Gilje Jaatun[1], Astrid Undheim[2] and Humberto Castejon[2]
[1]*SINTEF ICT, Trondheim, Norway*
[2]*Telenor Research and Future Studies, Trondheim, Norway*

Keywords: Cloud Computing, Security, Requirements, SLAs, Contracts, Brokering.

Abstract: The uptake of Cloud computing is being hindered by the fact that not only are current Cloud SLAs written in natural language, but they also fail to cover security requirements. This paper 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 the different Cloud providers, where requirements are stated in deontic contract languages. We investigate the suitability of seven of those 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.

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 (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, 2009). Keeping these in-house is considered to be less of a risk since the organization 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 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 as possible and take 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. In this paper we

Meland P., Bernsmed K., Jaatun M., Undheim A. and Castejon H..
EXPRESSING CLOUD SECURITY REQUIREMENTS IN DEONTIC CONTRACT LANGUAGES.
DOI: 10.5220/0003977906380646
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CloudSecGov-2012), pages 638-646
ISBN: 978-989-8565-05-1

have a special focus on SLAs covering security requirements, since they are considered more difficult to measure and quantify compared to the traditional ones. As pointed out by (Dwivedi and Padmanabhuni, 2008), security is only represented through policies, and not enforced through SLAs, and minimal work has been done on security SLA representation and enactment.

This paper considers a Cloud broker model where the broker help negotiate and establish SLAs between Cloud customers and Cloud service providers, based on security requirements expressed by the customer. The 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 the different service providers, where requirements are stated in a *deontic contract language* ("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 provide examples using some of the existing approaches which may be applicable for Cloud computing, and investigate their suitability for the Cloud brokering model though a practical use case for a video streaming service.

The paper is organized as follows. In Section 2 we discuss the role of security in the context of Cloud service brokering. In Section 3 we describe a use case that utilizes a broker to negotiate security requirements, and Section 4 shows some of the deontic contract language dialects that can be used to specify security requirements in SLAs. We discuss these approaches and related work in Section 5, and conclude the paper in Section 6.

2 CLOUD SERVICE BROKERING

As the uptake of Cloud services increases, we see a growing number of SaaS, PaaS and IaaS offerings from a broad set of providers. A Cloud broker is thus foreseen to become an important part of the Cloud ecosystem, providing a one-stop shop for consumers, SMEs and enterprises that look for a complete Cloud service portfolio covering all their IT needs. This Cloud broker will act as an intermediary between the customers and Cloud service providers (see Figure 1). It will facilitate choosing the best service providers to fulfill the requirements of a customer based on optimization algorithms. In its simplest form, this can be translated into picking the cheapest IaaS provider(s) whose offerings cover a minimum set of functional requirements. For SaaS, functionality may be of higher importance, and only small variations in non-functional requirements will be present. A Cloud broker will have business relationships with a set of Cloud providers offering different types of services. It is expected that the broker will negotiate and sign an SLA with those 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 customer requirements may then be used to choose between individual Cloud providers. However, to achieve this vision, there remain quite a few challenges related to contract scenarios as presented by (Leff and Meyer, 2007):

- "Contract documents are created using word processing applications. These documents can't 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 above challenges could be solved by using deontic contract languages. However, the customers may not have the necessary skill or technical expertise to write a deontic contract that can be automatically processed to find the best suitable candidates, and technical contract writing tools have not been widely accepted (Finnegan et al., 2007). An important role of the Cloud broker is therefore to help the customer translate the requirements into a 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 breach of SLAs. Since there is currently no prevalent standard for expressing contract terms, the Cloud broker would in many cases need to be multilingual.

3 USE CASE: CLOUDYFILMS

CloudyFilms is a new startup planning to offer video streaming on demand with a pay-per-view business

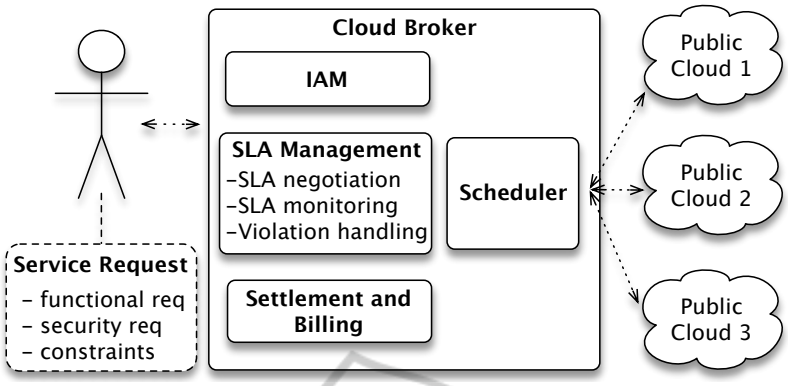

Figure 1: A Cloud broker model for ensuring secure Cloud deployment.

model. In order to minimize 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 to store 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.** A service provider may not subcontract processing or storage to a 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 on the customer registry's VM (VM2) to ensure they will always be hosted in European datacenters. 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 infrastructure/services from the other one, the latter becoming a single point of failure).

In this use case, there is not a single affordable 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 such a small company, CloudyFilms decides to use the services of a Cloud broker. The broker gets CloudyFilms' specification of hardware, security and QoS requirements, and uses it to select the cheapest IaaS providers that fulfill 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 US - see Figure 2). With each of the selected providers, the broker automatically negotiates an SLA on behalf of CloudyFilms and supervises its enforcement.

4 SPECIFYING SECURITY REQUIREMENTS IN SLAs

In this section we have selected a set of seven existing deontic contract languages found in the literature and for each of them tried to express at least one of the sample security requirements from Section 3. We have considered these languages based on the following properties:

- **Feasibility.** How well the language fits the type of requirements considered in the paper.
- **Complexity.** The size and expressiveness of the language.
- **Extensibility.** The possibility of adding additional concepts and expressions.

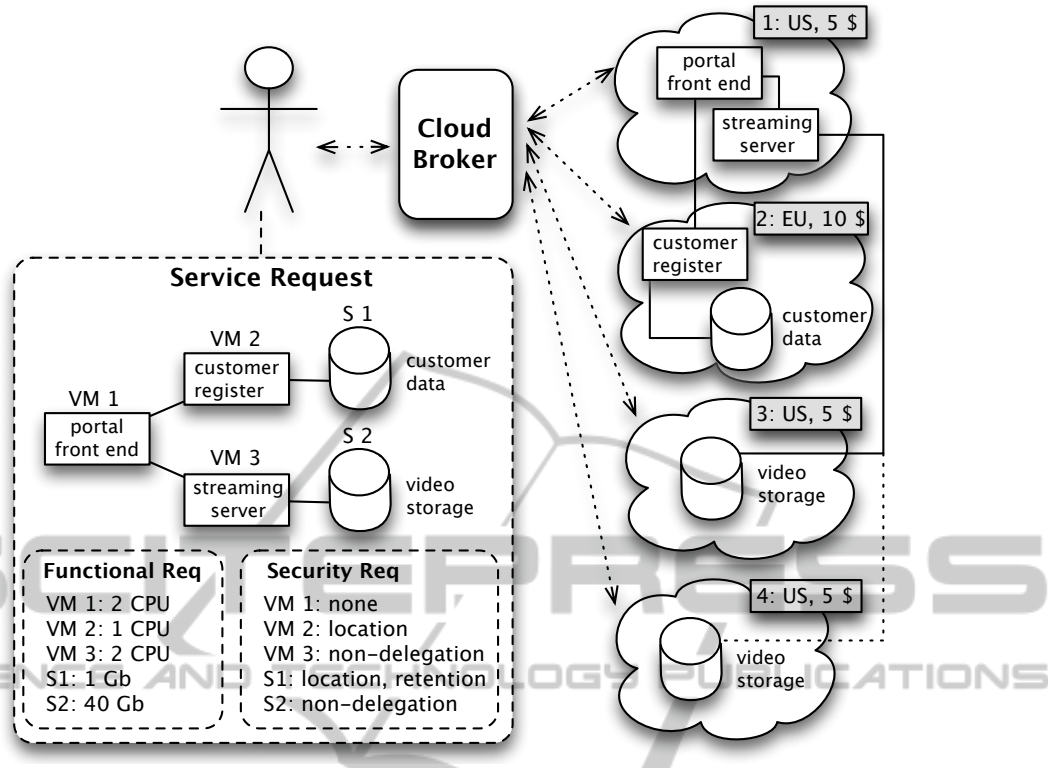

Figure 2: A Cloud Broker negotiating security requirements.

- **Maturity.** How long the language has been publicly available.
- **Support.** Associated documents, tools and other sources of information.
- **Adoption.** The current uptake among the relevant stakeholders.

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 given the context of service brokering for Cloud SLAs.

4.1 APPEL

The Platform for Privacy Preferences (P3P) specification (Cranor, 2003) allows 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 put restrictions on what data is collected by the provider and how it will be used (in terms of *purpose*, *recipient* and *retention*). The rule set in Listing 1 states that it is not acceptable that the service provider transfers the customer data to any third party (non-delegation). Here we have assumed that the customer data consists of physical and online contact information and purchase information.

With APPEL it is straightforward to put restrictions onto data storage retention by using the RETENTION element, however, it is not possible to articulate restrictions on the geographic location.

Writing policies in P3P/APPEL is fairly straightforward 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 privacy protection field. However, adoption was slow already from the beginning, and more recent numbers show that only a fraction of all websites have a P3P policy (Egelman et al., 2006). The work on the specification has been officially suspended since 2006.

Listing 1: An APPEL rule set for restricting access to customer data.

```
<appel:RULE behavior="block"
    description="Service may
    transfer customer data to 3rd
    parties">
<p3p:POLICY>
<p3p:STATEMENT>
 <p3p:DATA-GROUP><p3p:DATA>
 <p3p:CATEGORIES appel:connective="
     or">
  <p3p:physical/>
  <p3p:online/>
  <p3p:other-category/>
 </p3p:CATEGORIES>
 </p3p:DATA></p3p:DATA-GROUP>
 <p3p:RECIPIENT appel:connective="
     or">
  <p3p:same/>
  <p3p:other-recipient/>
  <p3p:public/>
  <p3p:delivery/>
  <p3p:unrelated/>
 </p3p:RECIPIENT>
</p3p:STATEMENT>
</p3p:POLICY>
</appel:RULE>
```

4.2 RFC 4745

RFC 4745 (Schulzrinne et al., 2007) is a framework for creating authorization 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 authorization 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, however, 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 2 will block service providers within the domains `example.com` and `example.org`. The purpose of RFC4745 was to increase interoperability by allowing authorization 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).

Listing 2: 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>
```

Listing 3: 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
```

4.3 XACML

XACML (*eXtensible Access Control Markup Language*) (OASIS, 2005) describes both a policy language and an access control decision request/response language. The typical setup is that someone wants to perform an action on a resource; however, XACML can also be used to find a policy that applies to a given request and evaluate the request against the policy.

The current version of XACML (2.0) 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 fulfills 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 3 (expressed in pseudocode to save space) states that customer data should not be stored by the provider or forwarded to any third party.

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

Listing 4: 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>
```

4.4 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 (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 on the service quality and/or resource availability offered by the service provider.

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 4.

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

4.5 Primelife Policy Language

The *PrimeLife Policy Language* (PPL) is an XML-based policy language developed by the EU project PrimeLife (PrimeLife Consortium, 2012). In addition to access control, PPL provides data handling as an extension to XACML 3.0 (Ragget, D. 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 and 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 5. In this data handling policy the AuthorizationsSet will be used to define what the service provider can do with the collected information and the ObligationsSet defines the obligations that the provider promises to adhere to. 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 5: Expressing non-delegation in PPL.

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

Listing 6: A data retention requirement expressed in BCL.

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

4.6 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 6 shows how one may express the data retention requirement as an *obligation* (pattern that must occur given an event) for the service provider:

Besides from obligations, modality can be used to express *permissions* (allowed behavior) and *prohibitions* (what must not occur). Given this expressiveness, all of our requirement examples should be

Listing 7: A non-delegation requirement in ConSpec.

```
SCOPE composition
SECURITY STATE
BEFORE invokeService(s)
PERFORM
   (s.non-delegation().equals(true))
      -> skip
```

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 seems therefore 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 behavior among the involved parties. Though BCL was introduced in 2005, there is currently little available information, tools and activities related to it.

4.7 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 triggers actions for updating the states. Listing 7 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 to others.

ConSpec is a relatively simple and restricted language, with a finite set of variables, no loops and intended for expressing security requirements. It is well suited for contract matching, can be somewhat extended, but is 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).

4.8 LegalXML

LegalXML is a standard from OASIS for structuring legal documents and information using XML and related technologies. A special technical committee developed *eContracts*(Leff and Meyer, 2007), which is an 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

Listing 8: An eContract 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>
```

processing systems rather than lawyers, and to structure any kind of contract. A simplified example representing the location requirement using eContract is shown in Listing 8. 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.

Though eContracts is a deontic contract language, the contractual terms within the blocks are still mostly defined using natural language. The main intention of eContracts is to represent contracts that have already been agreed upon and signed by the involved parties independent of word processing tools. This makes this approach 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 activity since to continue the work of eContracts.

5 DISCUSSION AND RELATED WORK

In the previous section we presented seven different specification languages and investigated to what degree it is possible to express the security requirements for the CloudyFilms use case. Table 1 shows a comparison of the languages based on the properties we defined in Section 4 using the values {*low, medium, high*}.

P3P represents early work in the context of pri-

Table 1: Comparing the specification languages in Section 4.

Language	Feasibility	Complexity	Extensibility	Maturity	Support	Adoption
P3P	Medium	Low	Medium	High	Medium	Medium
RFC4745	Low	Low	High	Low	Low	Medium
XACML	Low	Medium	High	High	High	High
WS-Agreement	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
LegalXML	Low	Low	High	High	Low	Low

vacy protection and usage of personal information, and several of the subsequently developed languages have been inspired by this work. However, now this initiative is more or less dead, mostly because the slow adoption and limited interest from the service providers. In our context the main limitation is data handling 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. WS-Agreement, BCL and ConSpec all seem suitable for expressing security requirements in a deontic form, however, neither specifies a security term ontology, leaving the problem of translating security requirements stated in natural language to a machine-readable format unsolved. 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.

Other languages that we have not explored in our Cloud broker use case include 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 they are 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). The Rule Based Service Level Agreement language (RBSLA) (Paschke, 2005) is another mark-up language for rule-based policy and contract specifications. Though there has been little activity from RBSLA since 2006, it was based on the still ongoing RuleML initiative (RuleML, 2012). A wider range of related policy languages and protocols can be found in papers by (Yagüe, 2006) and (Dwivedi and Padmanabhuni, 2008).

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 toward 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.

6 CONCLUSIONS

Current Cloud SLAs are written in natural language, and seldom cover security requirements; this is hindering the uptake of Cloud Computing. We envision Cloud brokers to have a role in translating customer requirements into deontic contract languages, thus helping users to find the most suitable Cloud providers in a more automated and dynamic way. Many different contract specification languages exist today, but 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 broking, and there is 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.*, 197:45–58.

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). https://forge.gridforum.org/projects/graap-wg/.

Aniketos Consortium (2012). Aniketos - ensuring trustworthinesss and security in service composition. http://www.aniketos.eu/.

Cranor, L. F. (2003). P3P: making privacy policies more useful. *Security & Privacy, IEEE*, 1(6):50 – 55.

Cranor, L. F., Langheinrich, M., and Marchiori, M. (2002). *A P3P Preference Exchange Language 1.0 (APPEL1.0)*. World Wide Web Consortium.

Dwivedi, V. and Padmanabhuni, S. (2008). Providing Web Services Security SLA Guarantees: Issues and Approaches. In Khan, K. M., editor, *Managing Web service quality : measuring outcomes and effectiveness*, chapter 13, pages 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, pages 197–207.

ENISA (2009). Cloud Computing: Benefits, risks and recommendations for information security. European Network and Information Security Agency.

Erlingsson, U. (2004). *The inlined reference monitor approach to security policy enforcement*. PhD thesis, Cornell University.

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*, pages 17–24.

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

Gartner (2011). Public Cloud Services, Worldwide and Regions, Industry Sectors, 2010-2015, 2011 Update. http://softwarestrategiesblog.com/2011/07/02/sizing-the-public-cloud-services-market/.

Governatori, G. and Milosevic, Z. (2006). A formal analysis of a business contract language. *Int. J. Cooperative Inf. Syst.*, 15(4):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*, volume 17 of *Communications in Computer and Information Science*, pages 221–236. Springer. 10.1007/978-3-540-88479-8_16.

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

Leff, L. and Meyer, P. (2007). eContracts Version 1.0. Technical report, OASIS. http://docs.oasis-open.org/legalxml-econtracts.

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.

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

OASIS (2005). *eXtensible Access Control Markup Language (XACML) Version 2.0*. OASIS Open. http:// docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

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*, volume 2, pages 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, pages 131–144. Springer.

PrimeLife Consortium (2012). Primelife - privacy and identity management in europe for life. http://www.primelife.eu/.

Ragget, D. et al (2009). H5.3.2 - Draft 2nd Design for Policy Languages and Protocols. Technical report, The PrimeLife project.

RuleML (2012). The Rule Markup Initiative. 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 http://tools.ietf.org/html/rfc4745.

Wang, X. (2010). Specifying the business collaboration framework in the Contract Expression Language. *International Journal of Business Process Integration and Management*, 4(3):200–208.

WSLA (2003). Web Service Level Agreements (WSLA) Project. 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*, 1(1):11–20.