On Incorporating Security Parameters in Service Level Agreements

Aida Čaušević, Elena Lisova, Mohammad Ashjaei and Syed Usman Ashgar

Mälardalen University, Västerås, Sweden

Keywords: Service Level Agreement, Security, SLAC, Security Process, Cloud Computing, Run-time Monitoring.

Abstract: With development of cloud computing new ways for easy, on-demand, Internet-based access to computing resources have emerged. In such context a Service Level Agreement (SLA) enables contractual agreements between service providers and users. Given an SLA, service users are able to establish trust in that the service outcome corresponds to what they have demanded during the service negotiation process. However, an SLA provides a limited support outside of basic Quality of Service (QoS) parameters, especially when it comes to security. We find security as an important factor to be included in adjusting an SLA according to user defined objectives. Incorporating it in an SLA is challenging due to difficulty to provide complete and quantifiable metrics, thus we propose to focus on a systematic way of addressing security using the security process.

In this paper we investigate ways in which security might be incorporated already in the service negotiation process and captured in an SLA. We propose a corresponding process to develop and maintain an SLA that considers both design-, and run-time. To demonstrate the approach we built upon the existing SLAC language and extend its syntax to support security. An example of a service being provided with security guarantees illustrates the concept.

1 INTRODUCTION

Cloud computing paradigm has provided new opportunities for resource sharing and increase of interoperability between different service providers. Given this, many organizations have noticed benefits with increased service efficiency, dynamic resource allocation and the cost decrease related to the system development and maintenance. However, such a business model with outsourcing as the key principle, introduces possible risks with regard to system security that is one of the vital attributes to be fulfilled in any system connected to the Internet.

Currently, to overcome security risks related to the cloud services, service users have to rely on a Service Level Agreements (SLA), a contractual agreement between a service provider and a service user established through the negotiation process, in order to assess service provider reliability, security policies and make an objective comparisons between different services offered. Beside cost and terms of use for every service, an SLA carries information regarding service attributes such as performance, availability, reliability. In most cases, service providers include into SLAs only those service attributes that is possible to measure and express using numeric values.

In the current practice, a cloud SLA focuses on a limited number of attributes, in most cases not including security, i.e., SLAs usually focus on performance with objectives related to availability (Jaatun et al., 2012). For the very few cases where security is taken into consideration, the way it has been addressed might be difficult to understand by service users (da Silva and de Geus, 2014; Casola et al., 2016a). Additionally, provided security guarantees are uniform for all provided services and users, regardless of particular service characteristics or specific user needs, in most cases boiling down to the availability attribute only. Also, security is still a non-negotiable attribute, meaning that there is no possibility to acquire a service with specific security characteristics (Petcu, 2014).

Many critical systems and applications are starting to introduce cloud solutions, as for such systems it is important to take security into account from the very beginning, clearly described security requirements combined with existing practice guidelines already at the design phase might lead to implementing...
the system with proper security in place. However, the actual level of system security in this case will also depend on the cloud services being involved. Additionally, to enable a way to deal with possible threats and risks that an adversary may impose, a security monitoring mechanism is required. Security monitoring in the cloud is a dynamic recurrent process which draws in the viable and dedicated administration of cloud segments to recognize and react to risks and threats on its services. One of the challenge regarding such monitoring is identification of relevant and effective parameters. However, once they are identified, based on the monitoring information service providers might require to refine provided guarantees and offer an updated or completely new SLA.

To address such challenges in this paper we focus on security considerations in SLAs with accounting for Confidentiality, Integrity, Availability (CIA) triad, where we first propose a process to develop an SLA taking into account security as one of the important properties to be considered. In our process we also address a run-time monitoring of services and effects of changes on existing SLA guarantees and possible re-negotiation process. Moreover, we extend the syntax of one of the existing SLA language, namely SLA language for Cloud Computing (SLAC), and provide an example that takes into account the proposed extensions.

The paper is organized as follows. Section 2 presents necessary background relevant for understanding the concepts of services, SLAs, SLA languages and security. Next, in Section 3 we present our approach on incorporation of security in an SLA where we describe the design-, and run-time process related to development and maintenance of an SLA along with the proposed extension of SLAC that enables the approach formalization. Furthermore, in Section 4 we present an example illustrating the approach via the extended version of SLAC. Relevant related work is described in Section 5, whereas conclusions with future work directions are presented in Section 6.

2 BACKGROUND

In the following we present preliminaries needed for understanding of proposed approach.

2.1 Services Terminology

A software service is a set of functions provided by a server software or system to a client software or system, usually accessible through an application programming interface (Broy et al., 2007). It can be created, invoked, composed and destroyed on demand. Services are developed to be platform independent and suitable for heterogeneous applications. Composite services can be built from the atomic ones with the main goal of a reusable functionality being provided by existing services in a low cost and rapid development process on demand. A composition can be achieved either through orchestration or choreography. The first assumes the existence of a central controller responsible for scheduling service execution according to the user demands, while the second assumes a mechanism of message exchange between participants in the composition, without requiring a central coordinator. A service interface provides information about specific service properties such as service type, capacity, time-to-serve, etc., visible to service users and used to find and invoke services most suitable for their needs. On the other hand, functionality representation is hidden from the service user and available only to service developers. Such a system development may be seen as a cost-efficient development by reusing functionality from available services. Also, a service becomes a single point of maintenance for a common functionality.

2.2 Service Level Agreements

One of the main challenges of using cloud technologies is that the quality of services cannot be controlled by cloud users. Therefore, the service qualities are negotiated and defined by both service users and service provider in an agreement, known as a Service Level Agreement (SLA) (Kyriazis, 2013). Given an SLA, service users are able to establish trust in that the service outcome is what they have demanded during the service negotiation process. An SLA contains a description of a service that carries information about various attributes, such as performance, availability ratio, reliability, etc. In most cases, an SLA includes only those attributes that can be expressed in terms of numerical values. On the other hand, security aspects of services cannot be presented as measurable number that makes them more difficult to be included and negotiated about. SLAs also contain a set of penalties that specify the regulations when the provider fails at delivering services at the agreed level of quality.

2.3 SLA Languages

In order to ease preparation and negotiation of an SLA several specialized languages are developed. SLA languages facilitate automating SLA
Table 1: An SLA example using SLAC.

| term group: Small_VM: | prv → cons:cCpu in [1,2] # |
| Cluster: prv → cons:RAM in [1,1] gb |
| terms: [1,1] of Cluster, prv → cons:replication is True |

negotiations and automatically creating SLA terms with their compositions. Among various SLA languages, the common ones include: Web SLA Language (WSLA) (Keller and Ludwig, 2003), Cloud SLA Language (CSLA) (Kouki and Ledoux, 2012), and a formal SLA language for Cloud Computing (SLAC) (Uriarte et al., 2014). A review of SLA specification languages has been done in (Maarouf et al., 2015), where the authors highlight the requirements, strengths and weaknesses of each language compared to each other. According to this work, all reviewed languages have formal syntax, however only CSLA and SLAC support formal semantics and formal verification. Moreover, SLAC provides support for incorporating brokers (by defining various parties), thus for this work we consider SLAC. In the following we explain the SLAC language syntax and structure in more details.

The core elements of the SLAC language include contract terms and definition of guarantees for terms. The contract terms specify the characteristics of a service given by a provider. Each agreement requires at least one term, which can be a metric or a group of terms. Further, a metric can be categorized by NumericMetric, which defines a range of values, BooleanMetric, which defines a 2-level value, and ListMetric that defines a list of values for a service. The SLAC language allows to define a set of terms in a group to be reused for several services. In this case the services should be clustered, e.g., a cluster of virtual machines with the same characteristics. In contrast to the terms, guarantees are not mandatory in an agreement. If a guarantee is defined in an SLA, in case of violation a penalty will be enforced to the service provider. A guarantee can be referred to a particular term or to any term in the agreement. When a violation occurs a set of conditions is evaluated and a set of actions can be taken. In such case, the conditionAction and actions should be defined in the SLA. The full semantics of the language can be found in (Uriarte et al., 2014).

Table 1 illustrates an example of an SLA using the SLAC language adopted from (Uriarte, 2015). In the assumed example, prv is a service provider, while cons is a service user. A virtual machine (VM) is defined in the agreement with two CPUs and 1 Gb RAM, which are defined by cons:cCput and cons:RAM respectively. Then, two of the VMs are clustered with a specific term on response time delay RT_delay within [0,0.6]ms. Finally, a term is defined for the cluster, which means it applies on all VMs in the cluster, that enables the replication. Note that in SLAC any term is defined in a range and the unit after the range, e.g., [1,1] gb means 1 Gb, while [1,2] # means between 1 and 2 units.

2.4 Security Terminology

Security can be defined as a system property that allows the system “to perform its mission or critical functions despite risks posed by threats” (Kissel, 2013), where a threat is defined as “the potential source of an adverse event” (Kissel, 2013).

In every system there is a set of assets, i.e., values that need to be protected against a malicious adversary. A vulnerability is described as a flaw in the system that enables a threat targeting one of the system assets. An attack is realization of a threat by exploiting a vulnerability in an attempt to break a system asset as it is demonstrated in Figure 1. Countermeasures are “actions, devices, procedures, or techniques that meet or oppose (i.e., counters) a threat, a vulnerability, or an attack by eliminating or preventing it” (Kissel, 2013). One can classify them as (i) preventive, e.g., encryption, (ii) detective, e.g., intrusion detection systems, (iii) responsive, e.g., blacklisting of a detected attack source (Miede et al., 2010). Countermeasures support security objectives, e.g., confidentiality and authentication.

Security process is a continuous process and it can be split into following steps (Kizza, 2017): (i) system security policy formulation, (ii) a security requirements elicitation, (iii) a threat identification, (iv) a threat analysis, (v) a vulnerability identification and assessment, (vi) a security certification, (vii) a security monitoring and auditing. A security policy can be defined as a set of policies and procedures that regu-
lates actions of people and systems within the information system security domain (Lopes et al., 2017). A policy can be evaluated for violations and enforced by mechanisms, it states how a high level security goal is achieved. Applied to a system level, two main groups of policies can be identified (McDaniel, 2005): (i) provisioning policies that prescribe a configuration meeting system requirements; (ii) authorization policies that map entities and resources into allowable actions. The latter is split into authentication policies responsible for stating how an identity of an entity can be established, and access control policies mapping the established identity to a set of corresponding rights.

3 SECURITY CONSIDERATIONS IN SLAs

SLAs include a set of attributes and guarantees on them, all of which are negotiated between a service provider and a user. These attributes can be quantifiable, e.g., a bandwidth, availability in terms of a guaranteed up-time, and non-quantifiable, e.g., safety, security. For the first group, it is straightforward to provide guarantees in terms of numbers, i.e., define ranges and conditions under which the attribute is guaranteed to be kept. However, providing guarantees for the second group is more challenging. This work is focused on a non-quantifiable attribute, namely security, that cannot be straightforwardly assessed by a number or by the fact that a security mechanism is being in place.

Security has to be addressed systematically, where particular solutions support corresponding security objectives. For example, an encryption protocol under assumptions about adequacy of its implementation and adequate usage, supports data confidentiality. However, the fact that encryption is correctly implemented and used, does not say anything about system security or even data confidentiality, as key handling and how data is stored at a server, have to be considered as well for assessing the security level. Given the complexity of assessing a service security level, it is a challenge to present it in a comprehensive, structured and clear way to a service user. Hence, we propose to divide assessment of the security process required for the further service negotiation into two parts: (i) assessment by a third independent party focused on a systematic side of the process, adequacy of analyses conducted, solutions, policies; (ii) assessment by a user, who trusts in the assessment conducted by the third parties and assess, i.e., negotiates, only particular security solutions, as they may contribute to different levels of security and imply different costs.

In the following we describe the process of incorporating security during design-time, and monitoring and maintaining it at run-time.

3.1 Security process for SLAs

In this work we consider security being incorporated in an SLA already at the design-time as it is one of the most important attributes to be considered in systems being connected to the Internet (i.e., left-hand side in Figure 2). Moreover, as security is dynamic by its nature, we consider run-time monitoring of services and effects of changes on existing SLA guarantees and possible re-negotiation process, as well (i.e., right-hand side in Figure 2).

Figure 2 presents the adopted security process (Kizza, 2017) embedded in the SLA development process divided into steps from 1 to 10. The combination of process allows to address security in a systematic way and develop an SLA for a service security level. Block 1 includes specification of a service, e.g., its functional specification, definition of required resources and connections. Block 2 contains application assumptions capturing possible instantiations of service functionalities for the assumed application and possible user requirements. Once the service and its specification are available, in Block 3 a security analysis of the service is performed, i.e., relevant security goals are specified, threat and vulnerabilities analyses are conducted.

Based on the analysis results security goals are translated into corresponding requirements. We skip the implicit step of requirements elicitation and instead, as more relevant for guarantees, present security policies and security mechanisms to be implemented, in Block 4 and Block 5 correspondingly. The term security policy is defined in Section 2.4. We choose to separate policies and mechanisms in two blocks as we want to make sure that the detailed technical specification of a particular mechanism mentioned in policies is captured in the right way, and both further are connected to guarantees.

The security process presented in Blocks 1-5 is supported by argumentation over adequacy of security level of the service. Further, mechanisms and policies are translated into guarantees, see Block 6, which contains assertions of what is guaranteed, under which conditions and with which possible following actions (e.g., to update, patch, maintain (Russo, 2018)). These guarantees later on are formalized by an SLA language into SLAs in Block 7. We assume Blocks 1-7 to present the design phase of the service and thus SLAs developed at this stage are the
 ones used for negotiation before the service being provided.

The rest of blocks (in bold) are used at run-time, i.e., during the service operation. Security mechanisms have to be monitored at run-time, see Block 10, to maintain an acceptable confidence level in their adequacy. Security policies also have to be assessed at run-time, marked as Block 9. These two are separated in different activities as monitoring implies checking values of particular parameters, e.g., a real-time check of consumed bandwidth, while assessment in our case implies non-quantifiable check, e.g., that policies are followed. Both blocks provide input to Block 6, as guarantees has to be checked at run-time, as well, for identification of possible violations. Once there are violations or possible changes leading to future violations, Block 6 sends this information to Block 8, where a refinement is performed. Block 8 maps the occurred change to a required refinement of an SLA and sends this information to Block 7, where the refinement is applied, i.e., the required corresponding action is triggered.

Above we described the development of an SLA starting at the service design, upon detection at run-time a violation or a change leading to future violations, two ways of addressing it are possible. The first one includes an update of the SLA, i.e., the SLA stays with the same terms. The second option is that due to a significant change we have to renegotiate the SLA, thus basically we develop at run-time a new SLA, building on top of the existing one.

As mentioned, we propose to divide assessment of an SLA development process and an SLA itself between an independent trusted third party and the service user. Blocks 1-3 represent a part of the security process and can better be assessed by a security expert. Similarly, the quality of design and implementation of Blocks 9 and 10 can be more effectively assessed by a security expert. Users in their turn, can assess results of Blocks 4-6 captured with a specific language in SLAs formulated in Block 7. Note, that SLAs also capture what is monitored and assessed and provided guarantees, without assuming that a user can effectively assess the quality of monitoring, assessment mechanisms, and their implementation.

### 3.2 SLAC Extension

In this subsection we present how guarantees from Block 6 in Figure 2 are transformed into the corresponding SLAs in Block 7. The formalization is done using the SLAC language, which is briefly described in Section 2.3. However, the defined language does not support constructs to incorporate security into it, thus, we built upon the already defined syntax in SLAC and extend it further to introduce security considerations within the development process presented in Figure 2.

The first step towards extending SLAC to cover security considerations is to add the term objective. We propose to include possible security objectives in the syntax, e.g., Confidentiality, Integrity. The objective can further have various categorization, similar to the metrics as explained in the language syntax. For our purpose, we consider to enrich BooleanObjective to include security mechanisms, such as Access Control, Encryption, Log, Key Management, Integrity Check. We consider presence of those mechanisms and correctness via a binary scale without an intermediate step reflecting low confidence in the correctness of an operating mechanism. Thus, we propose to set the access confidence level in this parameters as Boolean, i.e., it is either correctly implemented and used or not. Finally, ManagementAction includes Patch, Update and Maintain, where Patch and Update enforce a notification to the user about the action being taken.

As it was described in Section 3 incorporation of security requires applying the systematic security process and an independent assessment provided by a third party of its results and the process itself. How-
ever, a service user has an opportunity to grasp a high-
light of the process by assessing security policies that complement security mechanisms. Thus, we propose to introduce a construct enabling a policy hierarchy and an indication which policies pool it is a part of, namely belong. Policies are abstract and formulated on a high level of details, thus they are often decomposed further into (sub)policies associated with a particular objectives.

4 EXAMPLE

To demonstrate how security considerations can be included in an SLA, we extend the example described in Section 2.3. The provided service is the same, i.e., computational resources via a virtual machine, however now we enrich the system description with related security requirements. For example, considering data confidentiality and integrity we can formulate the following requirements: Requirement 1 – communication between Cluster and a user is adequately secure, and Requirement 2 – access to Cluster is adequately secure. In both cases the term adequate can be defined, e.g. according to ISO/IEC 27017 (ISO, 2015).

Considering related policies, there are two main groups. The first one reflects that systematic part of the process has to be reviewed independently, thus it can be formulated via the following non-use case specific policies: Policy 1 – a security case is built in parallel with service development, where security case is defined as in (Weinstock et al., 2014) and Policy 2 – the security assurance case is assessed by a third party. The second group of policies is more service specific and relates to security mechanisms. In this work we do not consider all required policies but focus only on those that support data confidentiality and integrity, namely Policy 3 – the authorization policy, which includes how authentication and access control are addressed.

Table 2 describes how the considered example can be expressed by means of the proposed extension of the SLAC language in Section 3.2. In order to capture relevant security mechanisms, Small_VM properties (cCpu, RAM) are complemented by authorization which is further presented by authorization:authentication and authorization:accessControl. Note, these two objectives are defined via different types, i.e., authentication just needs to be in place (boolean), as there is no related (sub)policy, only a mechanism that can be associated, and accessControl has a corresponding policy that belongs to the pool of authorizationPolicy. This policy defines which rights are granted to a particular authenticated user and might depend on e.g., a user, current service status, time of the day, location from where access is requested. Additionally, Small_VM has an objective regarding a possible stored password, as the point of the example is just to illustrate the approach, we consider a simple password structure and do not consider a key management and distribution. Above we consider authorization for a particular virtual machine, additionally at the Cluster level we have an objective supporting secure data transmission, namely encryptionAES, i.e., support for data encryption by means of the Advanced Encryption Standard (AES) algorithm. Finally, in terms we specify that the encryption and authorization may be included in the contract and their inclusion is a part of negotiation. Note, that the assessment of the algorithm’s implementation, done by a third party, depends on an assessor and evidences provided in the corresponding security case, while negotiation is done only upon including or not including those security mechanisms.

Moreover, one can specify guarantees to define actions given that a violation for an existing SLA occurs. Within guarantees one have a possibility to enforce quality of the service or an action will be taken in case of violations. In Table 3 we choose to put guarantees on encryption being in place and accessControl defined based on policies in authorizationPolicy. In case of violation of such
Table 3: Guarantees for the SLA example in Table 2.

<table>
<thead>
<tr>
<th>Guarantees:</th>
</tr>
</thead>
<tbody>
<tr>
<td>On violation of any:</td>
</tr>
<tr>
<td>IF encryption FALSE</td>
</tr>
<tr>
<td>OR accessControl NOT belong to authorisationPolicy</td>
</tr>
<tr>
<td>update in 24 hour AND negotiate</td>
</tr>
<tr>
<td>ELSE</td>
</tr>
<tr>
<td>update in 24 hour AND bonus:</td>
</tr>
<tr>
<td>1 hour of ([1,1] of Cluster)</td>
</tr>
</tbody>
</table>

Guarantees a provider is obliged to provide an update action within specified time and renegotiate with user (update in 24 hour AND negotiate). In case of such agreement between provider and user, one could choose to charge penalties in terms of bonus (bonus: 1 hour of ([1,1] of Cluster)).

Making service provision more flexible especially with regard to security related violations, one can look into different degrees of violations. It may help to provide means to distinguish and correspondingly respond differently to e.g., regular patching in a maintenance manner and a specific attack. Considering handling of new vulnerabilities, one can look into a vulnerabilities classification, e.g., a Common Vulnerabilities Scoring System (CVSS) (Mell et al., 2006), and provide responses correspondingly. In this work we do not go further in fine tuning guarantees in regards to different degrees of violation, as we only illustrate the main concept of incorporating security.

5 RELATED WORK

Addressing elicitation of security requirements for SLAs, in cloud services domain, is still in evolving stages. Luna et al. (Luna Garcia et al., 2012) justify the need of cloud administration architectures which depend on security requirements in an SLA. To identify the security parameters, there are distinct international standardization and rules, which help in specifying a common classification of security controls with both specialized and non-specialized features of security, i.e., the ISO27002 standard (ISO/IEC 27002, 2013), the NIST Security Control Framework (NIST SP-800-53, 2013), and the Cloud Control Matrix (CCM) – from Cloud Security Alliance (Cloud Security Alliance, 2013). An SLA alone does not ensure that the predetermined characteristics are met, but rather it characterizes the necessary monitoring mechanisms and actions following up identified events. Service level monitoring and support for its run-time adaptation are as imperative as the specification of an SLA.

Different commercial cloud service providers usually have their own solution for service monitoring. Microsoft Azure Suite (Microsoft Corporation, 2010) provides Azure Fabric Controller, which observes and oversees the servers, and also assigns resources for applications. The Amazon AWS platform offers CloudWatch (Amazon Web Services, 2006), where an observing framework has been offered for the control of resources and application administration. Google App Engine (Google, 2008) offers different monitoring solutions, e.g., CloudStatus (Hyperic, 2008) by utilizing a different set of APIs. All above mentioned providers do not offer SLAs with specific security guarantees, yet. Currently, a user cannot customize security features required for an application as documents that describe security in general are available. Thus, a user is lacking a possibility to specify a required security level and assess the satisfaction of the requirements from the provider’s side.

Petcu et al. (Petcu, 2014) have investigated monitoring types, monitoring behavior and a level of monitoring in the cloud services context. Based on their findings, security monitoring can be executed: on-premises, on monitored Infrastructure as a Service (IaaS) or via Software as a Service (SaaS). Security Information and Event Management (SIEM) systems are used in the first two cases. In the first case, a SIEM system uses specific APIs to collect logs from servers, whereas in the second case, a SIEM system can be loaded directly into an IaaS. In the last case, a particular observation from the cloud services is gathered (if accessible) and handed over to a security service provider. Muñoz et al. (Muñoz et al., 2012) have identified two types of approaches for monitoring software assets that provide assurance of the behaviors of software elements, static and dynamic. Static approaches focus on checking the security at development time, usually in simulated environments, while the dynamic approaches such as monitoring, surveillance and other forms of runtime analyses focus on the observation of the run-time software behavior.

Based on work of Aceto et al. (Aceto et al., 2013), one can distinguish between high and low level monitoring. The former relates to information on the status of the virtual platform that is collected at the middleware, application and user layers by providers or users through platforms and services operated by themselves or by third parties. The latter relates to information collected by the providers and usually not exposed to the users, with focus on the status of the physical infrastructure of the whole cloud (e.g., servers and storage areas). Security monitoring can be considered within high-level monitoring, while in low-level monitoring, specific utilities for collecting information about security might be related to the...
hardware layer, workload, voltage, temperature, to the operating system, software vulnerabilities and bugs, etc.

Kaaniche et al. (Kaaniche et al., 2017) present an SLA monitoring system based on the rSLA framework for the security properties in cloud services. Their work also describes the whole lifecycle of an SLA along with addressed security issues. Emeakaroha et al. (Emeakaroha et al., 2012b), present the DeSVi architecture used to detect an SLA violation via resource monitoring. The fundamental segments of the architecture are the automatic VM deployer, in charge of the distribution of resources and for mapping of assignments; application deployer, in charge of the execution of client applications; and LoM2HiS framework that screens the execution of the applications and makes an interpretation of low-level metrics into high-level SLAs. In their work they focus on monitoring and detecting an SLA violation at cloud infrastructure level only, but do not consider the application layer. In the other work, Emeakaroha et al. (Emeakaroha et al., 2012a) propose an architecture for monitoring. SLA violation identification at the application provisioning layer in Clouds, named Cloud Application SLA Infringement Detection (CASViD). The focus is on the application-level monitor, which is equipped for monitoring application measurements at runtime to decide their asset utilization practices and execution. Casola et al. (Casola et al., 2016b) provide a methodology to guide development of catalog of security services. Using such a methodology a user can negotiate and monitor the security adequacy of security services. The use of this system is the premise to empower the automatic enforcement of Security-as-a-Service. Moreover, the authors have proposed an approach to tackle this issue by introducing a per-service SLA model, which entails the use of a “tailored” SLA for each service. Rojas et al. (Rojas et al., 2016) propose a framework that provides a management of cloud services using the information about security requirements defined by the SLA, automatically, during its entire lifecycle. Moreover, proposed mechanisms offer support of all the phases of the SLA lifecycle. The framework comprises two sides: the user side that provides the interface to communicate with the provider, and the provider side that contains the proposed framework and their reconciliation with the cloud services infrastructure.

Silva et al. (da Silva and de Geus, 2014) propose a methodology to support the concept where a user may choose security metrics of required services with continuous monitoring provided by the environment. Authors focus on automatic creation of an SLA with security metrics defined and the depiction of the monitoring process for the cloud service infrastructure. They monitor security through a range of values (0 – 4) and focus on the problem of managing intangible and unmeasurable numbers. Moreover, they provide a way to manage security levels (top-down view) that considers values for each security metric with its respective risk, Quality of Service (QoS) and impact. When compared to our approach, most of the above described contributions do not consider the specific security metrics being in place, i.e., encryption, authentication, authorization, etc., thus ways how security is expressed is limited. The closest work to ours is the one presented by Muñoz et al. (Muñoz et al., 2012). However, compared to that work we go one step further and in the process assume an external third-party security assessor to maintain the quality of the SLA design.

6 CONCLUSIONS

The current state of the art limits SLAs to consider only those service quality attributes that are possible to be expressed using quantifiable metrics, such as performance, reliability, etc. However, given the development of cloud computing and new ways services are provided to their users, we find it as an important task to enable non-quantifiable qualities, such as security and safety to be included in SLAs, as well. There are some works that have already put some effort in defining ways to include security into an SLA (Muñoz et al., 2012; Casola et al., 2016b; Kaaniche et al., 2017), mostly focusing on expressing security levels or security in terms of availability. However, we target to provide an SLA that includes a larger set of security objectives (i.e., in context of CIA triad) during the design-time such that user gets guarantees on what is provided with respect to security. Moreover, we need to enable an SLA monitoring, maintenance, and renegotiation in case a provider fails at providing the guaranteed level of security.

To enable this, we have first propose a process of SLA development and maintenance in order to be able to capture security-relevant information and provide guarantees on top of it. Secondly, we extend syntax of an already existing SLA language, SLAC, to include security objectives. Additionally, an example of a service being provided with security guarantees captured within an SLA is used to illustrate the concept.

In the future work, we aim at implementing the constructs into the existing tools. Moreover, we plan to investigate how to address a confidence level in security SLA and possible degrees of violations and their consequences.
ACKNOWLEDGEMENTS

This work is supported by the Serendipity project funded by SSF, the SAFSEC-CPS project funded by The Knowledge Foundation, the FiC project funded by SSF, and within the context of the XPRES framework.

REFERENCES


