MÄLARDALEN UNIVERSITY

SCHOOL OF INNOVATION, DESIGN AND ENGINEERING
VÄSTERÅS, SWEDEN

DVA501 Thesis for the Degree of Master of Science (120 credits) in Computer Science with Specialization in Software Engineering 30.0 credits

INCORPORATING SECURITY IN SERVICE LEVEL AGREEMENTS

Syed Usman Asghar
sar18009@student.mdh.se

Examiner: Marjan Sirjani
Mälardalen University, Västerås, Sweden

Supervisors: Aida Causevic, Elena Lisova
Mälardalen University, Västerås, Sweden

September 2, 2020

Abstract

Given that the information and the physical worlds are merging, threats to integrity, confidentiality, and availability of information on cyber-physical systems, cloud computing, and other information technology (IT) services create new challenges. The quantitative estimation of security measurements is an extensively troublesome issue, and the dynamic nature of security is a hindrance to the automation of risk assessment. Information technology services are being used increasingly across organizational boundaries. These services are based on particular infrastructures, i.e., servers, networking systems, and databases, as well as on operating systems and application software. A Service Level Agreement (SLA) is a formal binding agreement between a services provider and a customer of the service in the context of a particular service provision. It represents the functional and non-functional features and properties the customer anticipates from the service. An SLA also cites the disciplinary measures and penalties that can be applied in the case of a violation.

Present-day SLAs encompass general Quality of Services (QoS) requirements such as performance, availability, reliability, cost, and report handling. Currently, security requirements are not enforced through SLAs since they are considered more difficult to measure and quantify compared to other QoS requirements. The absence of affirmation and clarity for security requirements in SLAs, with the current lack of procedures to evaluate security, frequently results in customers being not able to evaluate a security level of services. Recently, security metrics are being added as an SLA parameter, but still, multiple technical and usability issues make its adaptation difficult.

An SLA can be expressed in specialized languages designed for facilitating SLA arrangement, automating SLA negotiation, and adjusting services consequently as indicated by SLA terms. The primary aim of the thesis is to devise a process that incorporates security metrics in an SLA, assesses and monitors the SLA at run-time, and refine and update the SLA in case of violation. SLAC (Service-Level-Agreement for Clouds) is a language for SLAs, and this work presents the extension of SLAC language syntax to address the security issues in the form of security metrics. Moreover, two strategies for arguing over a security level of service are presented. Finally, we argue over a security confidence level of service depending on the quantification of the metrics to understand the violations during the service lifecycle.

Table of Contents

1. Introduction

These days, Cloud Computing is generally used to convey benefits over the Internet for both specific and economic reasons. The quantity of cloud-based services has rapidly expanded, firmly in the most recent years, as is expanded the unpredictability of the foundations behind these services. To properly work and oversee such complex frameworks compelling and productive monitoring is continually required [21].

As described in [1], service providers are ones who provide the actual storage, computing power, and network resources to the service customers. Service providers administrate the security of the service regardless of the customer. The simple fundamental security ensures that offered services are adequate for private computing professionals. However apparently not satisfactory for small organizations or for openly financed associations overseeing, for instance, medicinal services and Personal Information (PI) data to be ensured with particular security and protection necessities. The real problem is that security measures are expensive to offer, monitor, and update, and service providers are reluctant to provide these benefits to every customer. Ideally, customers would be looking for a tailored security feature that they can acquire on-demand as part of the service which they are getting from the services providers.

In the information technology (IT) services context, a Service Level Agreement (SLA) is a formal binding agreement between the services provider and the services' customer in the context of a particular service provision. It represents the functional and non-functional features and properties a customer anticipates from the services. An SLA also cites the disciplinary measures and penalties that could apply in the case of a violation. Contemporary SLAs encompass general Quality of Services requirements such as availability and performance, and the reporting and violation handling. Security requirements are not enforced through SLAs since they are considered more difficult to measure and quantify compared to other QoS requirements. The absence of affirmation and clarity for security requirements in SLAs, alongside the current lack of procedures to evaluate security, frequently results in customers being not able to evaluate the security of the services. Recently, security metrics are being added as an SLA's parameter, but still, multiple technical and usability issues make its adaptation difficult [42].

Metric is a standard of measurement that characterizes the conditions and the guidelines for playing out the measurement and for understanding the aftereffects of measurement [23].

Security parameters of service are hard to quantify due to its dynamic nature. To the best of our knowledge, there is no standard to quantify security metrics in SLAs, in the context of cloud services. In this work, we are aiming to develop a process for quantifying the security metrics and add these security features to the SLA which is written in SLAC [34], a language for the definition of SLAs devised explicitly for the cloud domain. To quantify the security metrics, a formation of a confidence level in system security which can help to identify the security violation (if any) in a much clear way. Security metrics will be defined in an SLA by using the semantics of SLAC. We are aiming for an extension to SLAC language syntax by adding some semantics that is suitable to quantify the security metrics.

The thesis work is organized in the following way; in section 2, Background, we discussed the SLA, its generation, QoS terms, SLA's phases, and the idea of security in SLAs briefly. Security challenges in SLAs are also described in section 2. In section 3, the problems have been formulated, and the research questions have been presented. In section 4, a review of existing literature to find state of the art SLAs, security parameters in SLAs, and models or frameworks to incorporate security in SLAs. As a method, a survey was conducted to write the review. One can see the review is divided into three main parts. The first part shows the state of the art SLA practices in the cloud domain, the second part shows how the security challenges have been addressed in SLAs in recent years, and in the third part, some examples of applications or frameworks are given where security challenges have been addressed to some extent using the SLAs in cloud services platform. In section 5, a brief introduction of three languages in which SLAs are written is given, and this section also argues about that why SLAC is chosen as the language for the proposed SLA with security incorporated. In Section 6, the proposed approach for the incorporation of security in SLA is explained. This approach is further discussed two possible strategies. We also argue about developing a confidence level in system security in this section. Section 7 will present a conclusion.

2. Background

SLA fundamentally portrays two things: the distinctive Service Level Objective (SLO) in terms of values for Quality of Service (QoS) metrics and the penalties in case of violations of those objectives. QoS is the capacity of service to meet specific prerequisites for the various features of the service. A service can have functional and non-functional aspects related to QoS, e.g., Google Apps provides a set of services with functional aspects like create, update and share the documents and non-functional aspects of a service can be performance, availability, reliability, or cost [46]. Multiple QoS metrics may be taken into account to define each aspect. Along these lines, a QoS metric is a way to evaluate the service level as a feature of QoS. One may need a service level to accomplish a given objective that is the Service Level Objective (SLO). While getting a cloud service, the definition of QoS parameters is a primary requirement. In this manner, a Service Level Agreement (SLA) is a set of SLOs that ought to be fulfilled and negotiated between the service provider and the customer.

Usually, SLAs address the non-functional requirements of the services expressed as service level objectives (SLOs). For example, Amazon's service availability is 99.95%, and it is mentioned on their site [42]. Security is the non-functional property of a service which cannot be expressed by a numeric value. The most mainstream services providers, for instance, Amazon and Google still don't offer SLAs including security-related guarantee terms however just report the security parameters their services accompany, by conceivably giving some specific details on their execution [23].

Custom security Service Level Objectives (SLOs) can be included in SLAs which are related to security metrics of that service. To address the concerns identified with security in cloud services, SLAs are these days updated to have security necessities which are often called Security SLA [40]. To implement these security-related regulations, monitoring of the service at the runtime is crucial. There are some tools available to monitor the performance parameters of the SLA [25] but not many of them address the security. To understand the role of SLA in cloud computing and how the security SLOs can be addressed in an SLA, one must know about the life cycle of a simple SLA.

2.1 SLA lifecycle management

SLA management is an extensive topic to study including numerous stages, to be a specific negotiation, deployment, monitoring, reporting, and termination phases as appeared in Figure 1. A similar approach is discussed by Rojas et al. in [6] and by Oktadini et al. in [16]. A brief review of each phase of the life cycle is described in the following paragraphs.

- Phase 1. *Definition and Specification*: This is the starting phase of SLA auto-administration, in which a formalized QoS demonstrate should be characterized and shared by both services provider and benefits customer to distribute their SLA offers and SLA requirements. This stage is portrayed by the determination of prerequisites to be incorporated into the SLA and also the recognizable proof of the consumer needs and qualities of the adopted service model. This topic is covered comprehensively by the TM Forum [3], and one or more of these exercises exacerbate it: initial specifications, template specifications, service development, publication, design and development, the definition of parameters and prices for offer and request, service offering, determine service provider, define SLA, service, and SLA template development. The obligation in this stage is shared between the service provider and consumer regarding definition and specification.

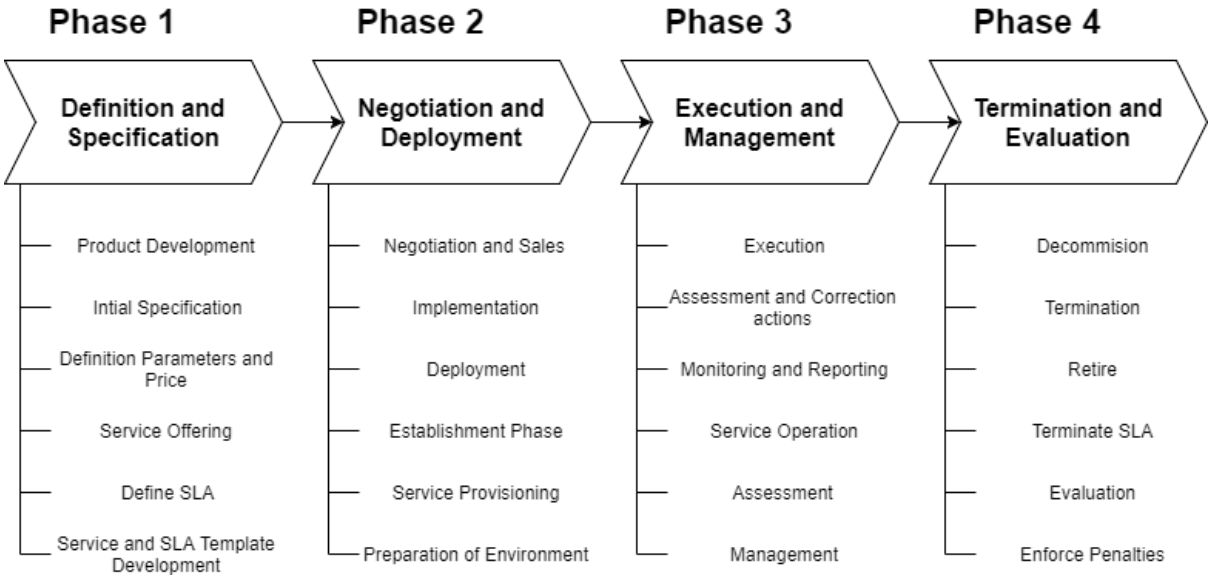

Figure 1 An SLA lifecycle (Redraw from [6])

- Phase 2. *Negotiation and Deploy*: The second phase of the SLA administration lifecycle means to set up the SLA contract for two parties, the service provider and the service

consumer, with the business relationship around specific web benefit provisioning. There are diverse business goals to be concerned for each party; the consumers expect that they can get the vital redistributing capacity as well as steady administration nature of their application with a limited expense, on the opposite end, the service provider wants to get amplified benefits through the service provisioning with limited assets expenditure. In the middle of the two closures, the SLA contract communicates the agreement on various terms with which both parties concurred for the foundation and institution of the business relationship. After the negotiation, the financial conditions and adequate levels of administration are characterized by the service provider and the consumer. Also, penalties are characterized by the two parties if there should be an occurrence of a violation of concurred conditions. Recurrence and substance of reports to be conveyed by the specialist organization are likewise characterized. This stage is aggravated by one or more activities: negotiation and sales, implementation, deployment, the establishment of phases, optimization of resource selections, distribution, service provisions, and infrastructural preparations. In this stage, the undertaking of the negotiation's obligation is shared between the service provider and the consumer but the service provider is exclusively in charge of the deployment tasks.

- Phase 3. *Execution and Management*: This stage is made out of administrations that are in operational consistency with the predetermined prerequisites of the SLA. After the resources are deployed, its monitoring is essential. It is difficult to monitor all the resources all the time, so random nodes of resources would be monitored for different sets of time. This monitoring would generate a massive amount of data and processing this data require additional resources and time so selected, and important data would be processed. Violation cautions that speak to the probability of an undertaking coming up short or then again not meeting its characterized service levels are once in a while announced as a feature of observed information. The objective is to go for predefined, steady choices in light of these alarms. The choice ought to be opportune and appropriate for the setting of the violation cautions. This stage is aggravated by one or more activities: execution, assessment and correction actions, monitoring, reporting, service operations, assessment, and management. The service provider is exclusively in charge of the whole phase undertaking.
- Phase 4. *Termination/Evaluation*: The evaluation phase is related to the audit of activities during the period in which services were used. At the termination phase, the agreement termination procedure is performed as a result of either contract expiration, violation of the

legal binding, or consumer demand. The allocated resources would be released and client access and awards would be revoked. Billing and issuing of invoices for the resources consumed are treated. In the case of violation of any biding at the service provider end, discounts or other compensations can be offered to the consumer. One or more activities comprise this phase: retire, termination, terminate SLA, assessment, and enforce penalties. In this stage, the service provider is in charge of the end undertaking while the consumer is in charge of the evaluation.

Examples: In [52] Casola et al. use the SPECS framework which aims at designing and implementing a framework for the management of the whole service level agreement life cycle, intended to build applications (SPECS applications) whose security features are stated in and granted by a Security SLA. Ghumman [53] presents a structural specification for the automation of SLAs life cycle in cloud computing. In the negotiation phase, a time-efficient technique is cultivated for simultaneously negotiating with numerous CSPs. A distributed monitoring approach is used for services being utilized at single or various locations and it decreases the number of communications of SLA violations to a monitoring coordinator by eliminating the unnecessary one. Oktadini et al. [16] use DAMIC (Six Sigma) together with IT Infrastructure Library (ITIL) to design SLA's life cycle for improving the quality of IT service delivery and support. Six Sigma was created by Motorola and the core of Six Sigma is a five-phase process improvement methodology called DMAIC [54]. DMAIC is an acronym for the five phases of the model (Define opportunities, Measure performance, Analyse factors impacting performance, Improve performance, and Control performance). They made a mapping of SLA's life cycle to the DMAIC cycle from Six Sigma. Six Sigma adds benefits to ITIL and helps organizations to adopt best practices for service delivery by a quality process that ensure its success.

2.2 Security challenges in SLA

Security issues of cloud computing are viewed as similar as in the ICT context, the need to address security in SLA for cloud setting was proposed in recent years. From fundamental security services Confidentiality, Integrity, and Availability (CIA Triad) [49] only availability requirements are addressed more comprehensively in an SLA, generally [6]. Even though security is an unmeasurable quality [47], usually security metrics are utilized to survey the security condition of a domain. Security metrics are instruments that give precise and current

data about the security condition of a domain, considering an assessment of activities and security controls in their condition [48].

The present landscape of practices, commitments, suggestions, and advantages identified with tending to the security requirements in the SLA by the services providers and consumers were overviewed by Rojas [6]. Through the study, the accompanying difficulties in cloud SLA were recognized: the design for overseeing security in SLAs, characterizing quantitative security metrics and not just qualitative metrics, SLA portrayal addressing security challenges, and security service declaration. Also, it was checked that the security viewpoints had been ignored in SLA contracts in regard to the requirements particulars and its related metrics. Besides, cloud services providers do not have characterized forms for dealing with the security prerequisites characterized in the SLA [7].

The accompanying practices have been recognized concerning the security necessities in SLA for cloud computing regarding the service providers [17][18][19].

- The service provider is exclusively in charge of deciding the event of security SLA infringement.
- The SLA is made out of conventional proclamations that educate how the provider must secure client information, without indicating the level of security and how this assurance is dealt with.
- Non-exclusive security estimations are made and put away by the service provider.
- There is no required notice of how the provider performs security estimations.
- The truancy of definitions and commitments to the security of client information.
- There is no determination of security service levels in the agreement, along these lines keeping the client from making appropriate choices on security conditions.

To incorporate security in SLAs, these can be the main issues: distinguishing proof of security metric's violation, evaluation of service in terms of security level, and persistent checking of security metrics. Concerning the identification of security metrics, a few rules, and international standardization activities exist, which are looking for characterizing a mutual list of security controls (e.g., ISO27002[8], NIST system [9], CSA's Cloud Control Matrix [10]) with the end goal to empower programmed transaction of security.

According to Cloud Security Alliance Guide, there are 14 best practices to ensure the security of cloud computing [55]: (1) Cloud Computing Architectural Framework, (2) Governance and

Enterprise Risk Management, (3) Contracts and Electronic Discovery, (4) Compliance and Audit Management, (5) Information Management and Data Security, (6) Interoperability and Portability Operating in the Cloud, (7) Traditional Security, Business Continuity, and Disaster Recovery, (8) Data Centre Operations, (9) Incident Response, (10) Application Security, (11) Encryption and Key Management, (12)Identity, Entitlement, and Access Management, (13) Virtualization, (14) Security as a Service.

3. Problem Formulation

The idea of a Security Service Level Agreement to determine the prerequisites of security services for an endeavor was first proposed by Henning [41] in 1999 and has been broadly received from that point forward to distinguish a legally binding understanding between a service provider and a service customer which expressly contains terms identified with security properties. In 2011, European Union Agency for Network and Information Security (ENISA) distributed a report breaking down the utilization of security parameters in cloud SLAs (for the most part engaged on the European Commission (EC) open division) [12]. The report indicated that, even though security was considered by most respondents as a top concern, they usually focus on just accessibility and other performance-related parameters, ignoring security-related ones. ENISA's Information Assurance Framework [13], based on ISO 27001/2 standards, cites directions for organizations to assess the risk related to the adoption of cloud services and to compare different offers with respect to security properties.

Even though cloud computing creases secure, the clients are concerned about the security of cloud computing. The Cloud Security Collusion (CSA) [14] [15] has distributed records concerning the top threats and security direction in distributed computing entitled "Top Threats to Cloud Computing." It is critical to feature that the two customers and providers are in charge of cloud security and the security necessities can differ as per the utilized service and deployment models. These viewpoints must be considered in an SLA definition, plus an SLA must consider the customers' needs and the current threats in this field. The Cloud Security Alliance (CSA) distributed an archive with the most noteworthy threats in a cloud environment. A customer should be able to choose the security aspects of the acquired services during the negotiation phase of the SLA lifecycle. The dynamic nature of security also demands a run-time SLA assessment and adaptation in the case of violation.

3.1 Research questions

In the context of the above-described challenges, this thesis aims at answering the following research questions:

RQ1: How to extend an SLA language such that it includes security metrics?

RQ2: How strategies of handling security breaches can be correlated with SLAs and whether they can be built upon a confidence level for system security?

3.2 Methodology

To answer these research questions, this thesis' work comprises of the following parts: A literature survey, as the selected method, would provide an understanding of the fundamentals and state-of-the-art literature about the research questions. As a result of this survey, we review the existing literature for SLA for IT services, state of the art security incorporation in SLAs, existing work to address the security challenges in SLAs, and languages that are in use to write an SLA. A mechanism of bibliographic review is used which enables us to find a general vision about the research topic. In this review, the selected studies can be classified into three main categories. The survey is led by a protocol that can be recreated and refreshed. This procedure contains three stages: planning, execution, and summarization.

The research protocol was made in the planning stage. This protocol contains; keywords, source databases, and choice and rejection criteria. The criteria are connected to the filters to choose just important outcomes. These filters are implied after the primary data is collected from databases. Then the information is extracted from the collected data. The search string was a combination of the keywords SLA, security, and cloud**.** The search was performed in the IEEE Xplorer, Science Direct, and SpringerLink. The key terms were searched in the titles of the articles and the timeline was set from 2012 to 2018. Two filters were applied to the gathered data, to sort out those works which would be productive to answer the research questions. The first filter was at the title level where we selected those papers which are related to cloud services, and the second at an abstract level where we select those papers which are propagating security issues in an SLA. In the summarization stage, we dissected the chosen publication to implement the characterization of the works and to examine the research questions.

We propose a process for security incorporation in SLAs and its management. This process shows the whole management of an SLA lifecycle and explains the runtime assessment and refinement of an SLA. We also extend the existing syntax of SLAC language to include security metrics in it. These metrics can address the security aspects of an SLA. An example of an SLA in SLAC language with security metrics is also given. This part covers the RQ1. To answer the RQ2, we discuss two possible strategies to argue over the security level of service. As a test case confidentiality and integrity aspects of the service are used. In the last, a discussion on the understanding of a confidence level in system security is given.

4. Related Work

As a result of the literature review, 30 publications were chosen for full-text reading and extracting the knowledge to answer the research questions. These publications are categorized into three groups according to its contents:

Group 1: General SLA, its phases, and general metrics
Group 2: Incorporating security in SLA, its phases, security metrics, state of the art security in SLA, and negotiations.
Group 3: Security based SLA model, services, and frameworks

4.1 General SLA, phases, metrics, violations, and negotiations

Serrano et al. [42] formally define Quality of Service (QoS) as the capacity of service to meet certain prerequisites for the various features of the service like performance, availability, reliability, or cost. To evaluate the QoS aspects of quantitatively and qualitatively, several metrics are used. A QoS metric is an intent to measure the service level concerning a QoS feature since the customer might need a service level to get a set goal, i.e., the Service Level Objective (SLO). SLA fundamentally portrays two things: the distinctive Service Level Objective (SLO) regarding values for Quality of Service metrics and the penalties in case of violations of those objectives. In this manner, a Service Level Agreement (SLA) is a set of SLOs that ought to be fulfilled and negotiated between the service provider and the customer [42]. Xiong et al. [11] describe the general SLA metrics, e.g., performance, and availability requirements categorize them and explain that how these metrics are based on QoS requirements which the cloud service provider promises to offer. They study the computing resources allocation analyze audit logs for ensuring the guarantee of an SLA.

Rojas et al. [6] propagate in detail about SLA life management. They explain how an SLA lifecycle can be divided into different phases and also give a brief look upon the merits and constraints of each phase. They also show an investigation between the security necessities and the SLA lifecycle, went for understanding the conveyance of these prerequisites all through the periods of the SLA lifecycle to show if facilitate security advancement is required. Emeakarohaa et al. present the detection of SLA violation infrastructure (DeSVi) architecture, sensing SLA violations through sophisticated resource monitoring [20]. The core parts of the DeSVi architecture are: (i) the automatic VM deployer, (ii) application deployer, and (iii) the

LoM2HiS framework. In light of client request for, the programmed VM deployer allocates vital resources for the asked service and orchestrates its sending on a virtual machine (VM). After service deploying, the LoM2HiS framework screens the VMs and deciphers the low-level metrics into high-level SLAs as per user requirements having used the Domain-Specific Languages (DSLs). To acknowledge autonomic SLA administration DeSVi uses a learning database for the assessment of the observed data with the end goal to propose receptive activities if there should be an occurrence of SLA violation.

4.2 State of the art security in SLA, phases and security metrics

Rojas et al. [4] present the proposition of an SLA lifecycle identified with cloud computing and their particular stages and ideas identified with the security requirements of SLA for cloud services. State of the art of issues identified with the utilization of security requirements in the SLA setting for cloud computing. The actual practices, commitments, and proposals for security SLA were briefly discussed. They also present the comparison between the security requirements and the SLA lifecycle, went for understanding the dissemination of these requirements all through the periods of the SLA lifecycle to demonstrate if facilitate security improvement as it is required. Torkura et al. [42] analyze OpenStack's (a cloud operating system) [44] security using vulnerability density, vulnerability lifecycle metrics, patching trends, and vulnerability severity. Based on this analysis they defined three main security metrics, with classifications, for their work. The first security metric, *Vulnerability Lifecycle*, alludes to the period from when a vulnerability is found in software until when it is disposed of entirely. Software vulnerabilities experience the following stages: *Discovery Time*, *Exploit Time*, *Disclosure Time*, and *Patch Time.* The second security metric is *Risks Involved in Vulnerability Lifecycles,* and these risks are: *Black Risk*, the risk between discovery time and disclosure time; *Grey Risk*, appears between disclosure time and patch time; *White Risk*, after the release of a vulnerability risk, the software could even now be misused if not yet patched by clients. The third security metric, *Vulnerability Density*, measures the number of vulnerabilities per line of code.

Carvalho et al. [21] provide a systematic mapping of the literature about research that addresses the security in SLAs. The main objectives of the research are to find the consideration of security in SLAs, definitions, and measurement of security parameters in SLAs, negotiation of the security parameters, and the features observed in SLA operationalization. Results show that

an important number of studies focus only on the operationalization. They discovered a few threats related to this condition. The referred threats are shared technology vulnerabilities, insufficient due diligence, data breaches or data loss, and Denial of Service (DoS). Given the current threats, it is fundamental to characterize a vocabulary that communicates the SLA expressions and their association with the security controls. SLA negotiation focuses on processes for selection of services or providers, i.e., propose a method to select a provider in accordance, or the provider can present an SLA template to be used by clients as support for negotiation.

4.3 Security based SLA model, services, and frameworks

Petcu et al. [22] provider's establishment or as an outer administration. It is an independent segment of a bigger framework that permits negotiation of service level agreements and their enforcement. This module empowers the clients to ceaselessly watch out for their applications concerning certain security properties that may hold any importance to them. The module screens the resources and services and informs occasions considered of importance (as indicated by the SLAs) to an implementation module. The module incorporates existing and custom observing instruments/operators to assemble data on SLO metrics, to help the implementation module to distinguish conceivable alarms and violations. Casola et al. [23] described a methodology to set-up a catalog of security capabilities that can be offered as-a-Service, on top of which specific guarantees can be specified through a Security SLA. They Illustrate the SPECS approach to cloud security through SLA management with particular focus on the construction of secure supply chains. Expose metrics and enforceable parameters to customers instead of service levels, which can be chosen by customers as desired service level objectives.

Takahashi [45] proposes a mechanism to assemble a non-repudiable security service level agreement (SSLA) between a customer and a service provider. This article proposes a system to fabricate non-repudiable service level agreements, between a client and a service supplier. The proposed technique characterizes three jobs: User, Service Provider (SP), and Knowledge Base (KB). To develop an SSLA, the security prerequisites and abilities must be plainly expressed. "Security prerequisites" is data that composes what sort of security or security innovation is required, and "abilities" is data that composes what sort of advances the SP has. The proposed strategy has given four measurements: Target, Risk, Function, and Technique,

so different clients can uninhibitedly depict the security prerequisites and capacities. The proposed strategy can develop an SSLA that guarantees non-reputability by utilizing a security expression technique, ID conversion technique, negotiation protocol, and SSLA determining algorithm.

Analysis

Security is considered a real test for the endorsement of cloud services on an extensive scale acceptance in consumers. Addressing the security specifications and the definitions of security requirements in an SLA, in the cloud services domain, is still in evolving stages. An SLA alone does not ensure that the predetermined characteristics are met, but rather it characterizes the necessary monitoring mechanisms. Service level monitoring is as imperative as the specification of an SLA. Different commercial cloud services providers usually provide their solution for monitoring the services [56]. Microsoft Windows Azure Suite [57] provides Azure Fabric Controller which observe and oversee the servers also plan resources for the applications. The Amazon AWS platform offers CloudWatch [58] an observing framework offered as a benefit for the control of resources and application administrations. Google App Engine [59] offers the use of different monitoring solution i.e. CloudStatus [60] by utilizing a different set of APIs.

These prominent cloud services providers still do not offer SLAs which have specific security guarantees. The customer does not have a choice to choose such services which comes with specific required security features. They only document the security features of the services with some technical details about the exertion. The client, along these lines, needs to acknowledge the administration for what it's worth and is commonly not furnished with any confirmation identified with the dimension of security-related with the service [23]. Most of the above-described contributions do not consider the specific security metrics being in place, i.e., encryption, authentication, authorization, etc. which shows that security is expressed in a limited capacity.

5. Languages for SLAs

In the result of the literature survey, we also find the publication in which languages for SLA are described and their features. In the following, three languages will be described with its main features, and a comparison would be drawn between these languages. These languages are Web Service Level Agreement (WSLA), Cloud Service Level Agreement (CSLA), and Service-Level-Agreement Language for Cloud Computing (SLAC).

5.1 Web Service Level Agreement (WSLA)

IBM proposed the Web Service Level Agreement (WSLA) language [27]. The WSLA language is an XML (eXtensible Markup Language) based language; it is characterized as an XML schema. WSLA consists of an XML-based definition language and an architecture that supports the SLA lifecycle. WSLA is a hierarchical language to define SLA parameters from resources through business objectives. In a sample hierarchy structure of WSLA, SLA parameters are the topmost elements. Also, an SLA parameter is made from at least one composite metric, and each composite metric is the aggregation of some resource metrics [27]. WSLA can be utilized by both service providers and service customers to design their frameworks to give and regulate their services. This procedure is defined as deployment. Every association utilizes its deployment functions that decipher the WSLA and conduct the applicable activity. Furthermore, a WSLA defines which party monitors the service, the third parties which are used to measure the metrics, and supervision of guarantees or violation management. Cooperation among the parties managing the WSLA is likewise characterized [28].

A given SLA in WSLA consists of three basic concepts: parties, service description, and service obligations [27].

- The parties section identifies all the contractual parties. Signatory Party descriptions contain the identification and the technical properties of the parties, i.e., their interface definition (e.g., the way they accept events) and their addresses. The definitions of the Supporting Parties contain, in addition to the information contained in the signatory party descriptions, an attribute indicating the sponsor(s) of the party.
- The service description section of the SLA specifies the characteristics of the service and its observable parameters.

- The obligations part defines various guarantees and constraints that may be imposed on SLA parameters.

Keller and Ludwig [27] defined a WSLA framework for defining and monitoring SLAs between service providers and service consumers. This framework works well in the electronic commerce types of scenarios that involve two parties with distinct roles; the service provider is offering a service, and the service consumer is requesting and consuming the service. However, in the context of dynamic collaborations, this does not work well due to the unique characteristics of dynamic collaborations [30].

5.2 Cloud Service Level Agreement (CSLA)

Cloud Service Level Agreement (CSLA) [30] language describes QoS-oriented SLA associated with cloud services to allow SLA management strategies to be more flexible and elastic. CSLA presents a novel approach that accommodates features about QoS uncertainty and cloud fluctuations. These features can be listed as confidence, penalty, and ambiguity. The amount of ambiguity sets the acceptable limits for the target value of SLO. The confidence level is the percentage of consistency of SLO provisions and penalties are applied in case of SLA violations, to compensate services' customers. CSLA models the SLA violation penalty as a linear function: $P = \alpha + \beta dt$ where β is the penalty rate, and dt is delay time [30]. Figure 2 describes the main elements of CSLA expressed in a UML class diagram.

In Figure 2, the CLOUDSLA class has an instance that specifies the SLA. This instance is encompassing *parties* and *obligations*. A *party* (generally a cloud service provider) executes a few functionalities which can be expressed as at least one *service*. A party may depend on services given by different parties, which is shown by *references* to those services. *Obligations* characterize the connection between a service and a reference. Obligations depend on the *guarantees*, i.e. SLOs. A guarantee consists of requirements, confidence, and a penalty. A guarantee can be expressed in two ways either in a simple expression or by a composite expression. The combination of guarantees is finished utilizing the arrangement of *operators* characterized in the class *Operator*. Once an SLA is portrayed with CSLA and set up between a service provider and a service customer, it is delivered to the online cloud controller framework [30].

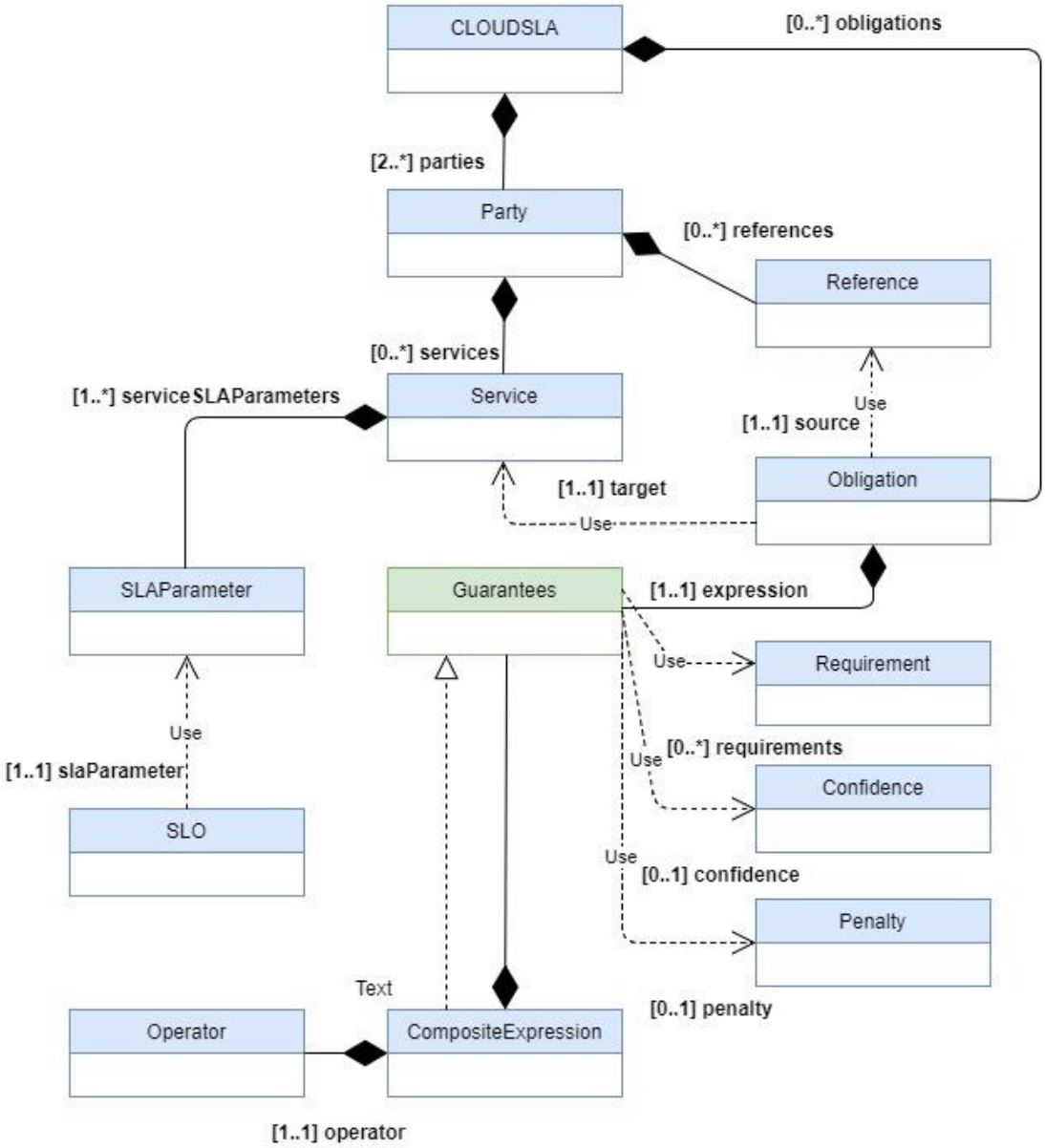

Figure 2. CSLA meta-model [30]

5.3 Service-Level-Agreement Language for Cloud Computing (SLAC)

Service level agreement language for Cloud Computing (SLAC) is devised for specifying an SLA for the cloud computing domain. The formal ground of SLAC is the SLAC Management Framework [34] which is free, open-source software developed for supporting the specification, evaluation, and enforcement of SLAs in cloud systems. The main elements of an SLA are the description of the contract, the specification of terms, and the definition of the

guarantees for these terms. This core-language gives the fundamental elements that empower the description of a basic SLA for the cloud computing domain. Comparatively to the majority of the current SLA definition languages, SLAC does not separate service description terms and quality requirements. Table 1. shows the formal definition of the syntax of the core language, which is defined in the Extended Backus Naur Form (EBNF) [33].

SLA	::=	`id:` *Id* `parties:` *PartyDef PartyDef*$^{+}$ *Expiration* `term groups:` *Group** `terms:` *Term*$^{+}$ `guarantees:` *Guarantee**
Expiration	::=	`valid from:` *Date* `expiration date:` *Date*
PartyDef	::=	*PartyName*$^{?}$ `roles:` *Role*$^{+}$
Role	::=	`consumer` | `provider` | `carrier` | `auditor` | `broker`
Group	::=	*GroupName* : *Term*$^{+}$
Term	::=	*Party* `->` *Party*$^{+}$: *Metric* | `[`*Expr*`,` *Expr*`]` `of` *GroupName*
Party	::=	*Role* | *PartyName*
Metric	::=	*NumericMetric* `not`$^{?}$ `in` *Interval Unit* | *BooleanMetric* `is` *Boolean*
	|	*ListMetric* `has` `not`$^{?}$ {*ListElement*$^{+}$} `or` {*ListElement*$^{+}$}*
NumericMetric	::=	`cCPU` | `RT_delay` | `response_time` | `RAM` | `availability` | `jitter` | ...
Interval	::=	`]`*Expr*, *Expr*`[` | `]`*Expr*, *Expr*`]` | `[`*Expr*, *Expr*`[` | `[`*Expr*, *Expr*`]`
Expr	::=	*Literal* | `infty` | *NumericMetric*`(`*Parameter*`)` | *GroupName* | *Expr Operator Expr*
Parameter	::=	`min` | `max`
Operator	::=	`+` | `-` | `*` | `/` | `>` | `>=` | `<` | `<=` | `and` | `or`
Unit	::=	`gb` | `mb/s` | `ms/min` | `minute` | `seconds` | `ms` | `month` | ...
BooleanMetric	::=	`back_up` | `replication` | `data_encryption` | ...
ListMetric	::=	`operating_systems` | `jurisdiction` | `hypervisor` | ...
ListElement	::=	`occi` | `ec2` | `kvm` | `xen` | ...
Guarantee	::=	`on` *Event* `of` (*Party* `=>` *Party*$^{+}$`:`)$^{?}$ *GuaranteeMetric* : *ConditionAction*
GuaranteeMetric	::=	(*GroupName*`:`)* *NumericMetrics* | (*GroupName*`:`)* *ListMetrics*
	|	(*GroupName*`:`)* *BooleanMetric* | (*GroupName*`:`)* *GroupName* | `any`
Event	::=	`violation`
ConditionAction	::=	(`if` *Expr* `then` *Action*$^{+}$)$^{+}$ (`else` *Action*$^{+}$)$^{?}$ | *Action*$^{+}$
Action	::=	(*Party* `=>` *Party*$^{+}$`:`)$^{?}$ *ManagementAction*
ManagementAction	::=	`notify` | `renegotiate`

Table 1. The syntax of the SLAC language (copied from [33])

In the SLAC language, the parties included ought to be characterized in each term, for example, the party liable to satisfy the term (a single party) and the customer of the service (one or more). This categorical definition adds to help multi-party agreements, to lessen ambiguity in the mapping of the monitoring responsibilities. Besides, it may be utilized for enhancing the security aspects of the agreement, for example, the incorporation with the authorization control. This control allows just those parties to have access which is engaged with the terms [33]. The

metrics accessible in the language are pre-characterized in the light of the necessities of the cloud domain.

5.4 Comparison between SLAC and the existing languages

Most cloud computing providers offer SLAs as a text description, and shift the burden of monitoring and enforcing the SLA to the customers [34]. However, several solutions were proposed in the literature to define and enforce a machine-readable SLA. The most well-known are: SLAng [35], a domain-specific language for IT services; and WSOL (Web Service Offerings Language) [36], WSLA [37] WSAgreement (Web services agreement specification) [38] and SLA* (An abstract syntax for Service Level Agreements) [39], for general-purpose services.

		WSOL	SLAng	WSLA	WS-Agreement	SLA*	CSLA	SLAC
General	Cloud Domain	-	-	-	-	□	■	■
	Cloud Service Models	-	-	-	-	□	■	□
	Multi-Party	-	-	-	-	-	■	■
Business	Business Metrics	□	□	□	□	□	■	■
	Price schemes	□	-	□	□	□	■	■
Formal	Syntax	■	■	■	■	■	■	■
	Semantics	-	□	-	-	-	■	■
	Verification	-	□	-	-	-	■	■
Tools	Evaluation	■	■	■	■	■	■	■
	Open-Source	■	■	-	■	■	■	■

Table 2 Comparison of the SLA languages. [33]

Table 2 shows the comparison between SLAC and those languages which are used for the definition of SLAs. Where ■ represents a feature covered in the language, □ partially covered feature, and - when no support is provided. The chosen highlights are identified with the extent of generic arrangements but not in a conclusive way. As we are using SLAC, with security semantics as its extension, as the language for the SLA in this work so we compare SLAC as a language here in Table 2.

The results of the comparison among the language show that the existing languages do not support some of the main requirements of the domain. Among the significant gaps in the definition of SLAs, there is a lack of support for multi-parties, cloud domain, cloud service models, business metrics, and vocabulary of the domain and the formal specification of the SLA evaluation. 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 chapter, we explain the SLAC language syntax and structure in more details SLAC is offering most of the requirements and possibilities to include security metrics as the extension of the language. The research question RQ1 is answered in this section and languages for SLAs are briefly described. A short comparison is drawn between these languages to advocate the choice of SLAC to describe an SLA and the extended version of SLAC.

6. Our approach

In this thesis work, we are proposing a process through which we can incorporate security in an SLA already at the design time. Security needs continuous overseeing due to its dynamic nature, so we consider run-time monitoring of services and effects of changes on existing SLA guarantees and possible re-negotiation process and run-time assessment of security policies. These policies can be set during the evaluation and negotiation phase of the SLA. Figure 3 is a graphical depiction of the process. Blocks in single line boundaries are part of service design/development and blocks with bold boundaries are part of service operation. While blocks with double boundary participate in both functions, service design/development, and service operation. The combination of the process allows us to address security systematically and develop an SLA for a service security level. This process will be explained in the context of the SLA lifecycle's phases.

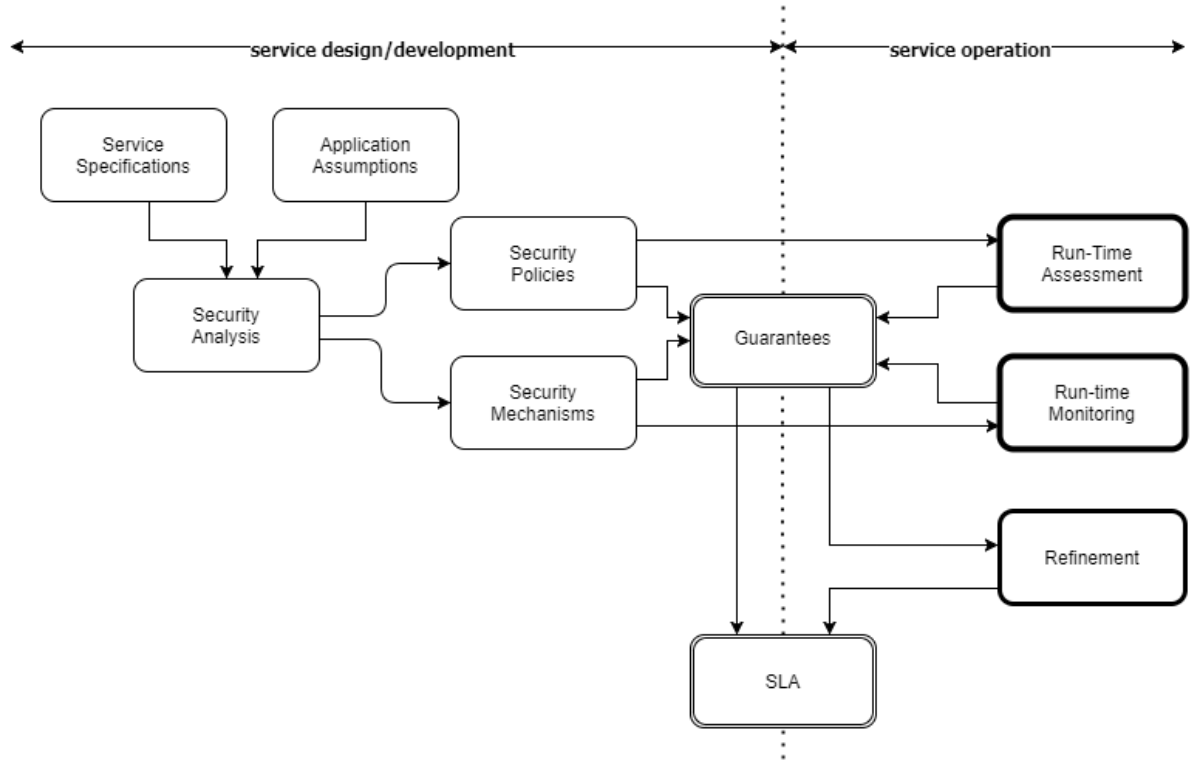

Figure 3. Security incorporation process for an SLA and its management

In the *service specifications* block, both parties share their offers and requirements of the services. e.g., its functional specification, the definition of required resources, and connections. Block *application assumptions* deal with possible instantiations of service functionalities for the assumed application and possible user requirements. In Figure 3, these tasks are shown as

part of service design/development. SLAs, mostly, are written automatically in a machine-readable format upon the selection of service by the customer. The customer has the option to choose from in the form of different QoS aspects. Service specifications and Application assumptions are those tasks that the customer and service provider perform at the definition and specification phase (phase 1) of the SLA lifecycle (Figure 1). Once the service and its specification are available, *security analysis* block conduct a security analysis of the service to find out the risks, vulnerabilities, and threats related to that services. This information leads to set the security goals for this service. These goals are translated into a set of policies and procedures that regulates the actions of customer and service provider. This operation takes place in block *security policies* (see Figure 3). Detailed technical specification of a particular mechanism, mentioned in policies, is captured in the *security mechanisms* block. All these tasks are the parts of the negotiation and deployment phase (phase 2) of SLA's life cycle (Figure 1).

The security process introduced in blocks *service specifications, application assumptions, security analysis, security policies, and security mechanisms* are upheld by argumentation over-sufficiency of the security level of the service. Further, mechanisms and policies are converted into guarantees, see block *guarantees,* which contains affirmations of what is ensured, under which conditions and with which conceivable subsequent activities (e.g., to update, patch, maintain). These guarantees subsequently are formalized by an SLA language into SLAs in block *SLA* (see Figure 3.)

Guarantees, and *SLA* blocks in Figure 3, can be the part of execution and management phase (phase 3) (Figure 1) of SLA's lifecycle and can also be part of the evaluation phase. Policies and security mechanisms will be translated into the SLA's guarantees, and this will complete the process of SLA generation. After the negotiations between concerned parties, the service can be launched. Run-time assessment of security policies and run-time monitoring of the security mechanism is the execution and management phase (Figure 1). Run-time assessment of policies allows updating the guarantees in the case of any changes in service from either customer or service provider. This would allow readjusting the SLA's objectives. Run-time monitoring of security mechanisms will lead to an adequate confidence level in the system in the context of security.

These two blocks are separated in different activities as *run-time monitoring* implies checking values of particular parameters, e.g., a real-time check of consumed bandwidth, while *run-time assessment* in our case implies non-quantifiable check, e.g., that policies are followed. Both blocks provide input to block *guarantee*, as guarantees have to be checked at run-time, as well, for identification of possible violations. Once violations or possible changes are leading to future violations, block *guarantees* send this information to block *refinement*, where a refinement is performed, block *refinement* maps the happened change to a necessary refinement of an SLA and sends this data to block *SLA*, where the refinement is applied, i.e., the required corresponding action is set off.

As described earlier, the development of an SLA starts at the design of the service. In the case of detection at run-time, a violation, or a change that is leading to a future violation two scenarios can opt. 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.

One of the main challenges of using cloud technologies is that the quality of services cannot be controlled by cloud users. Therefore, service qualities are negotiated. 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.

6.1 An extension on SLAC language with security constructs

The formal ground of SLAC is at the basis of a software framework [34] developed for supporting the specification, evaluation, and enforcement of SLAs in cloud systems. The SLAC Management Framework is free, open-source software [32]. However, the existing syntax of the language does not support constructs to incorporate security into it, so we built upon the already defined syntax in SLAC and extend it further to introduce security considerations for the security incorporation into the SLA process as shown in Figure 3. The general syntax of the language is shown in Table 1. Table 3 lists the names of the metrics and their corresponding extensions which covers the security considerations.

Term		Extensions
Objective	::=	Confidentiality | Integrity | ….
NumericMetric	::=	…. |Recovery time|
BooleanObjective	::=	….|Access control| Encryption| Log| Key management| Integrity check|
ManagementAction	::=	….|Patch| Update| Maintain

Table 3. Extended syntax of SLAC for security considerations

Each SLA requires the definition of at least one term, which can be either a Metric or a Group of terms. A metric can be of three types: (i) *NumericMetric*, which is constrained by open or closed Intervals of values (that can be defined explicitly in the SLA or inferred from the evaluation of expression) and a particular unit (e.g. milliseconds, gigabytes); (ii) BooleanMetric, which can assume true or false values; and (iii) *ListMetric*, whose values are in a list. We add a term *objective* as an extension and include *Confidentiality*, *Integrity* in the syntax to achieve the security targets. The *objective* can further have various categorizations, similar to the *metrics*. *BooleanObjective* is used to included security mechanisms, such as *Access Control*, *Encryption*, *Log*, *Key Management*, *Integrity Check*. It is considered that 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 parameter as Boolean, i.e., it is either correctly implemented and used or not. *Numeric metric* has an extension in the form of *Recovery time*; it shows the time that a system needs to recover from an attack. Finally, *Management Action* includes *Patch* and *Update* where Patch and Update enforce a notification to the user about the action being taken.

An Example

To demonstrate how security considerations can be included in an SLA, we are presenting an example. This example is adapted from SLAC Management Framework [32] and Uriarte et al.[33].

SLA

```
Id: 1234
```

```
parties:
        IDT
        role: provider
        SYED
        role: consumer
valid from: 28/08/2018
expiration date: 31/01/2019

term groups:
Tiny_VM:
        IDT->SYED : cCpu in [1,2]#
        IDT->SYED : RAM in [1,1] gB
        IDT->SYED : authorization : authentication is Ture
        IDT->SYED : password has {char [12]}, {INT, SYMB}
Cluster:
        IDT->SYED : RT_delay in [0.0 , 0.6]ms
        IDT->SYED : encryption in [128 , 256]b
        IDT->SYED : recovery_time [min*, max*]min
                              *pre-defined in service

        [2,2] of Tiny_VM
        terms:
        [1,1] of Cluster
        IDT->SYED : replication is True
        IDT->SYED : Log is True
        IDT->SYED : authorization is True
        IDT->SYED : encryption is True
```

Constraints:

```
#SLA Terms
1 ≤Cluster≤1 ^
IDT,SYED : replication == True ^
IDT,SYED : log == True ^
IDT,SYED : authorization == True ^
IDT,SYED : encryption == True ^



#Constraints of the cluster group

0.0 ≤ Cluster:IDT,SYED:RT_delay:0 ≤ 0.6 ^
0.0 ≤ Cluster:IDT,SYED: encryption:128 ≤ 256 ^
```

```
      0.0 ≤ Cluster:IDT,SYED: recovery_time:min ≤ max ^
      2 ≤ Cluster:Tiny_VM ≤ 2

#Constraints of Tiny_VM, instantiated in
#Cluster Group
1 ≤ Cluster:Tiny_VM:IDT,SYED:cCpu:0:0 ≤ 2 ^
1 ≤ Cluster:Tiny_VM:IDT,SYED:RAM:0:0 ≤ 1 ^
1 ≤ Cluster:Tiny_VM:IDT,SYED:cCpu:1:0 ≤ 2 ^
1 ≤ Cluster:Tiny_VM:IDT,SYED:RAM:1:0 ≤ 2
```

In this SLA example, it is stated that the provider (IDT) must provide a *Cluster* to the consumer (SYED), support a list of interfaces and that the service must have the replication [32][33]. Services also have log, authorization, and encryption. The service named *Cluster* is a group of 2 VMs with a minimum of 1 and a maximum of 2 CPUs and 1 Gigabyte of RAM. Moreover, the response time between these machines must be lower than 0.6 milliseconds, the encryption technique is between 128 to 256 bits and recovery time must exceed the pre-defined maximum value in minutes.

6.2 Strategies for arguing over a security level of a service

Data security might come at a higher cost than expected for either administrative or economic reasons, but data integrity might have a higher price because a wrong decision or event can cost the organization a substantial sum of funds or potentially even a misfortune in market esteem. In Figure 4, we present a graphical depiction of cloud services with the runtime monitoring and assessment by the third party; an SLA expressed in SLAC language with runtime updates and renegotiations, an application which has the user interface to generate the SLA and handle the update and service provider administrative portal. In the context of our approach, a client can use the application to create the SLA or handle the updates through it. This application is also connected to the services provider administration portal so they can patch any changes in services or SLA's terms to the user. After the service acquisition, continuous monitoring will be started. A third party or even service provider, depending on the agreed terms, is responsible for runtime assessment and monitoring. If there is a breach in service, the monitoring system will notify the services provider administration and also notify the customer.

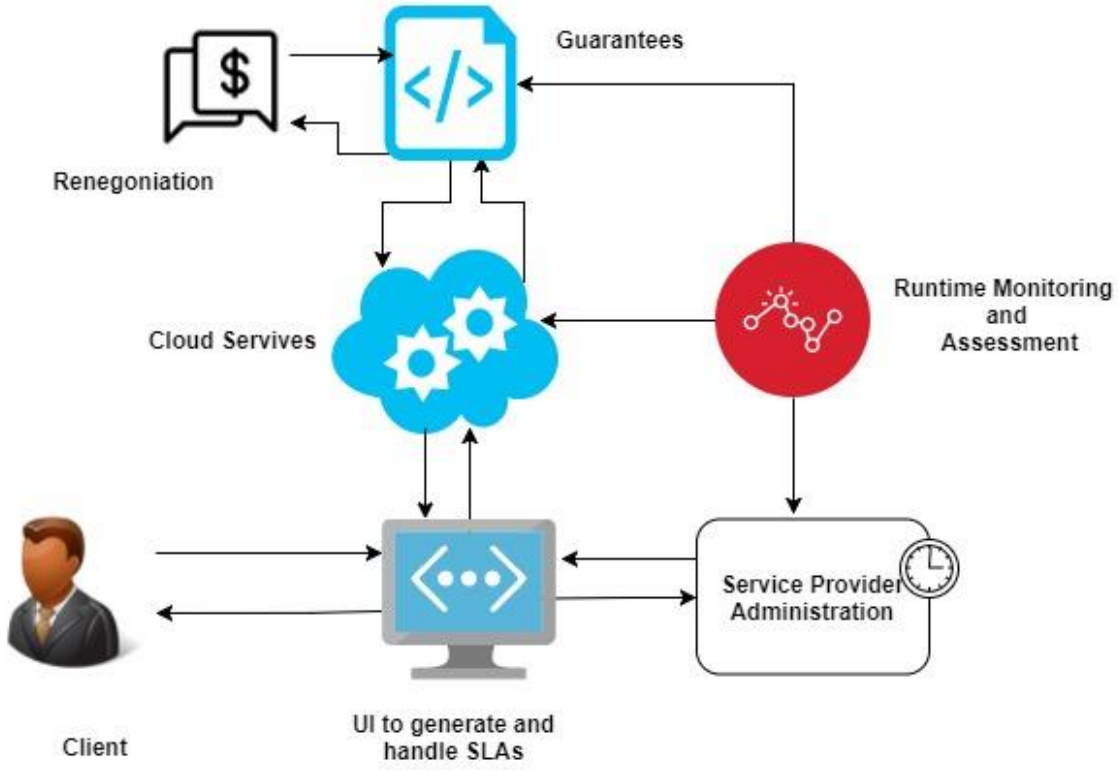

Figure 4. Service architecture for a security-driven SLA

Confidentiality, Integrity, Availability (CIA) triad principles [49] are network security terms standing for attacks and breaches in data confidentiality, integrity, and availability. As an SLA metric, availability is comprehensively addressed in general SLAs [6]. As an extension of the SLAC language, we include "confidentiality" and "integrity" as security metrics. This inclusion helps to quantify the security metrics. As proposed in extended syntax, a system either provides the “confidentiality” or not, and similarly “integrity” can also be expressed. To understand it in a better way, we can say that each metric can be evaluated as 0 or 1. System security, as a whole, has a value of 3 which sum of all individual values of each metric if the system is in a secure condition. So, if security is 3, it means all the properties are intact, and if it is less one or more it is a breach. This breach means that the confidence level of the service is reduced. This scenario should trigger the new guarantees and hence a new SLA. In the following two possible strategies are discussed in which we argue over a security level of service, based on the nature of the attack.

6.2.1 Strategy for the attacks on system confidentiality

In the context of information technology, confidentiality means that data should remain secret only for those who may have access to it. When dealing with a distributed services environment, confidentiality suggests that a client's data and computational undertakings are to be kept classified from both the cloud provider and different customers. Confidentiality stays as one of the main concerns with regards to distributed computing [51].

A security attack will be considered successful if any of the security metrics fail to hold its predefined values. Let's assume that customer signed the SLA with a provider and the service is launched, at this point the confidence level is 3 as we defined policies and their management correctly. The system is under attack, and it got successful. For example, it was a Cross-VM attack via Side Channels. Ristenpart et al. [50] explain the Cross-VM attacks in an Amazon EC2 platform. These attacks enable attackers to co-reside with another VM on the same physical machine and release the confidential data. This means the confidentiality is now 0 instead of 1 and the system confidence level is reduced. Now the security attack was successful due to an existing vulnerability in the system which the attack has exploited and caused the loss of one or more of CIA metrics.

To handle this or similar kind of attack, one can use the confidentiality, access control, log, and recovery time objectives (see Table 3) from the extended version of the SLAC language. “Confidentiality” objective has an attribute of “has” or “has not”, access control and log metrics have a Boolean attribute, and "recovery time" has the predefined value of time depending upon the service. An attack will be considered successful, in this scenario, if any of these metrics do not meet SLA terms. In this case, control measures which are the security policies (Figure 3) should be updated to handle this vulnerability (bugs or internal weakness of system) in the form of a new patch or security update. So then we accordingly update the values of metrics and confidence level as per the quantification mentioned above. This process is done in the “refinement” block (Figure 3), SLA's guarantees are also going to be updated according to this security patch, and hence a new/updated SLA can be produced on runtime.

6.2.2 Strategy for the attacks on system integrity

In the context of information technology, integrity means the accuracy, consistency, and reliability of the information content, processes, and systems. Prevention of the unapproved or unauthorized modification of the data also falls under the spectrum of integrity [51]. The integrity in distributed computing comprises both data integrity and computation integrity. Data integrity suggests that data ought to be genuinely put away on cloud servers, and any infringement (e.g., data is lost, modified, or traded off) are to be recognized. Computation integrity infers the idea that programs are executed without being contorted by malware, cloud provider, or different malicious clients, and that any erroneous registering will be identified.

Data loss/manipulation can be one example of attacks on integrity. Administration errors may cause data loss. To counter such or similar attacks, one can use theses security metrics “integrity” with the attribute “has or has not”, “integrity check”, and “encryption” with the attribute of Boolean”, and “recovery time” has the predefined value of time depending upon the service (see Table 3) Encryption metrics ensure the proved and authorized modification and “integrity check” enables the dependability. A similar approach, as discussed in the previous section" from the process (Figure 3) would hold in the case of any metric does not hold its pre-defined values.

These are the only two possible attacks that we discussed as they are considered good examples to explain the usage of extended metrics to address the security aspects of an SLA. Many more likely situations can be handled by including theses security metrics of different combinations. In the following section, a confidence level in system security is discussed which helps to understand the use of security metrics in SLAs.

6.3 Confidence level in system security

If an authority is to decide whether to accept a system based on its assurance argument is compelling or not, that authority would need to assess the confidence that the argument justifies and to use that assessment in a test of sufficiency. To access such confidence level, we need the quantification of all these security metrics which were mentioned in the strategies. Then their combination can be collective, we can argue here more on the importance of one metric or another and its contribution towards the combination or merely assign them a standard score. Let’s say multiply all of them by 1 which will have no effect but can be changed for some

providers depending upon the service they are providing and can be discussed in the negotiations phase, anyhow considered as the confidence level. Similarly, different metrics can be the outcome of certain metrics or combinations of different metrics.

For example, a system has the confidentiality if the access control, log is TRUE and recovery time is within its predefined value. Similarly, a system has integrity if the integrity check and encryption metrics are TURE. Different service providers have their specific amount of availability i.e. Amazon's service availability is 99.95% [42]. Also, depending upon the nature of the attack the recovery time, for example, can be low, medium, and high and could contribute to security confidence itself. One idea could be to think of the attacks for which the vulnerabilities are known and map it to the recovery time, for example, buffer overflow is a known vulnerability and if it is not handled and some attack exploits it to cause confidence reduction on security, the recovery time might be low. So, one can say that system has a confidence level of 3 as a whole whereas confidentiality has a value of 1, integrity has a value of 1, and availability has a value of 1 and with these different combinations, recovery time can be low, medium, and high.

Every customer has its own preferences based on the needs and the nature of the acquired services. For example, availability can be crucial for those customers who are using software systems as services, and for personal information handling, banking services, or similar capital involved transaction businesses prefer the confidentiality and integrity in the system. Having the confidence level is system security, a customer can choose independently and include such security metrics that are satisfying its needs.

7. Conclusions

Distributed services for computational tasks are taking the place of conventional computational infrastructures by every passing day. Security is the primary challenge for such systems to provide its services and adaption to these services. 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. Addressing security aspects in SLAs is an imperative topic for research in academics and industry to make it possible for the secure provision of these services.

In this thesis work, we have presented a process to incorporate the security in an SLA. In this process, we demonstrated the SLA development and management in such a way that the security aspects can be monitored at runtime. Based on this monitoring information, guarantees of an SLA can be updated and renegotiated. This process also illustrates how a runtime modification/updating of an SLA is possible. We also present the extension of SLAC syntax to address the security issues in the form of security metrics. We argue on the security confidence level of a service depends on the quantification of the metrics and how a customer can create its own level of confidence for a service by using this quantification. Two possible strategies for arguing over a security level of service are presented. It is also explained that how the quantification of security metrics would help to determine the security breach and what would be the possible process of refinement of an SLA in such a case. This extension of the SLAC language is in the early stages, and we aim to provide further extensions to include several other metrics.

8. References

[1] V. Casola, A. De Benedictis, M. Rak and U. Villano, "SLA-Based Secure Cloud Application Development: The SPECS Framework," 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, 2015, pp. 337-344, DOI: 10.1109/SYNASC.2015.59.

[2] F. Faniyi and R. Bahsoon, “A Systematic Review of Service Level Management in the Cloud,” ACM Comput. Surv., vol. 48, no. 3, pp. 1–27, Feb. 2016, DOI: 10.1145/2843890.

[3] G. Damm, G. Bain, J. Timms, L. Philippart, R. Roman, J. Deheus, A. Cruz, I. Best, D. Milham, and M. A. Alhakbani, “Gb917 – SLA management handbook version 3.0”, TeleManagement Forum, Tech. Rep., November 2012. [Online]. Available: http://www.dmtf.org/sites/default/files/standards/documents/DSP2029n%20 1.0.0a.pdf

[4] M. A. T. Rojas et al., “A framework to orchestrate security SLA lifecycle in cloud computing,” presented at the 2016 11th Iberian Conference on Information Systems and Technologies (CISTI), Jun. 2016, DOI: 10.1109/cisti.2016.7521372.

[5] J. Ding and Z. Zhao, “Towards autonomic SLA management: A review,” presented at the 2012 International Conference on Systems and Informatics (ICSAI), May 2012, DOI: 10.1109/icsai.2012.6223574.

[6] M. A. Torrez Rojas et al., “Inclusion of security requirements in SLA lifecycle management for cloud computing,” presented at the 2015 IEEE 2nd Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE), Aug. 2015, DOI: 10.1109/espre.2015.7330161.

[7] V. Casola, A. De Benedictis, and M. Rak, “Security Monitoring in the Cloud: An SLA-Based Approach,” presented at the 2015 10th International Conference on Availability, Reliability and Security (ARES), Aug. 2015, DOI: 10.1109/ares.2015.74.

[8] International Organization for Standardization, “ISO/IEC 27002:2013 Information Technology, Security Techniques, Code of Practice for Information Security Management,” 2013.

[9] National Institute of Standards and Technology, “NIST SP-800-53: Recommended Security Controls for Federal Information Systems,” 2013.

[10] Cloud Security Alliance, "Cloud Control Matrix v3.0," https://cloudsecurityalliance.org/download/cloud-controls-matrix-v3/.

[11] K. Xiong and X. Chen, "Ensuring Cloud Service Guarantees via Service Level Agreement (SLA)-Based Resource Allocation," presented at the 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops (ICDCSW), Jun. 2015, DOI: 10.1109/icdcsw.2015.18.

[12] M. Dekker and G. Hogben, "Survey and Analysis of Security Parameters in Cloud SLAs Across the European Public Sector," ENISA, Tech. Rep., December 2011.

[13] ENISA, "Procure Secure. A guide to monitoring of security service levels in cloud contracts," April 2012.

[14] Cloud Security Alliance, "Top Threats to cloud computing", Online: https://downloads.cloudsecurityalliance.org/assets/research/top-threats/treacherous-12-top-threats.pdf. Accessed: December 16, 2018

[15] Cloud Security Alliance, "Cloud computing vulnerability incidents: a statistical overview", Online: https://crow.org.nz/sites/default/files/2017-01/Cloud_Computing_Vulnerability_Incidents.pdf

[16] N. R. Oktadini and K. Surendro, "SLA in cloud computing: Improving SLA's life cycle applying six sigma," presented at the 2014 International Conference on Information Technology Systems and Innovation (ICITSI), Nov. 2014, DOI:10.1109/icitsi.2014.7048278.

[17] J. Meegan, G. Singh, S. Woodward, S. Venticinque, M. Rank, D. Harris, G. Murray, B. Di Mastirno, Y. Le Roux, J. McDonald, R. Kean, M. Edwards, D. Russel, and G. Malekkos, "Practical guide to cloud service level agreement," Cloud Standards Customer Council (CSCC), Tech. Rep., April 2012.
Online: http://www.cloudstandardscustomercouncil.org/2012 Practical Guide to Cloud SLAs.pdf

[18] C. Baudoin, J. Flynn, J. McDonald, J. Meegan, M. Salsburg, and S. Woodward, "Public cloud service agreements: What to expect and what to negotiate," Cloud Standards Customer Council (CSCC), Tech.Rep., March 2013.
[Online] http://www.cloud-council.org/publiccloudSLA.pdf

[19] J. Luna, H. Ghani, T. Vateva, and N. Suri, "Quantitative assessment of cloud security level agreements - a case study," in In Proceedings of the International Conference on Security and Cryptography, ser. SECRYPT 2012. SciTePress, 2012, pp. 64–73.

Available: http://www.deeds.informatik.tu-darmstadt.de/fileadmin/userupload/GROUP DEEDS/Publications/conf/secLA eval.pdf

[20] V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya, and C. A. F. De Rose, "Towards autonomic detection of SLA violations in Cloud infrastructures," Future Generation Computer Systems, vol. 28, no. 7, pp. 1017–1029, Jul. 2012, DOI: 10.1016/j.future.2011.08.018.

[21] C. A. B. de Carvalho, R. M. de C. Andrade, M. F. de Castro, E. F. Coutinho, and N. Agoulmine, "State of the art and challenges of security SLA for cloud computing," Computers & Electrical Engineering, vol. 59, pp. 141–152, Apr. 2017, DOI: 10.1016/j.compeleceng.2016.12.030.

[22] D. Petcu, S. Panica, B. Irimie, and G. Macariu, "On Security SLA-Based Monitoring as a Service," in Internet of Things. IoT Infrastructures, Springer International Publishing, 2016, pp. 326–336, DOI:10.1007/978-3-319-47063-4_34

[23] V. Casola, A. De Benedictis, M. Erascu, M. Rak, and U. Villano, "A Security SLA-driven Methodology to Set-Up Security Capabilities on Top of Cloud Services," presented at the 2016 10th International Conference on Complex, Intelligent and Software-Intensive Systems (CISIS), Jul. 2016, DOI: 10.1109/cisis.2016.116.

[24] NIST Special Publication 500-307, "Cloud Computing Service Metrics Description", NIST Cloud Computing Reference Architecture and Taxonomy Working Group, 2015, DOI: 10.6028/NIST.SP.307

[25] N. Kaaniche, M. Mohamed, M. Laurent, and H. Ludwig, "Security SLA Based Monitoring in Clouds," presented at the 2017 IEEE International Conference on Edge Computing (EDGE), Jun. 2017, DOI: 10.1109/ieee.edge.2017.20.

[26] Dana Petcu and Ciprian Craciun, "Towards a Security SLA-based Cloud Monitoring Service", CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, pp 598-603.

[27] A. Dan et al., "Web services on demand: WSLA-driven automated management," IBM Syst. J., vol. 43, no. 1, pp. 136–158, 2004, DOI: 10.1147/sj.431.0136

[28] A. Keller and H. Ludwig, Journal of Network and Systems Management, vol. 11, no. 1, pp. 57–81, 2003, DOI: 10.1023/a:1022445108617.

[29] H. Ludwig et al., "Web Service Level Agreement (WSLA) Language Specification", 2003, Publisher: IBM Corporation Inc.

[30] S. Nepal, J. Zic, and S. Chen, “WSLA+: Web Service Level Agreement Language for Collaborations,” presented at the 2008 IEEE International Conference on Services Computing (SCC), Jul. 2008, DOI: 10.1109/scc.2008.66.

[31] Yousri Kouki, Thomas Ledoux. “CSLA: A LANGUAGE FOR IMPROVING CLOUD SLA MANAGEMENT,” presented at the 2nd International Conference on Cloud Computing and Services Science, 2012, pp.586-591, DOI:10.5220/0003956405860591.

[32] SLAC Management Framework http://code.google.com/p/slac-language/

[33] R. B. Uriarte, F, Tiezzi, and R. De Nicola, “SLAC: A Formal Service-Level-Agreement Language for Cloud Computing”, 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp 420-426. DOI: 10.1109/UCC.2014.53

[34] P. Patel, A. Ranabahu, and A. Sheth, “Service Level Agreement in Cloud Computing,” in Cloud Workshops at OOPSLA09, 2009.

[35] D. D. Lamanna, J. Skene, and W. Emmerich, “SLAng: A Language for Defining Service Level Agreements,” in FTDCS. IEEE, 2003, pp. 100–106.

[36] V. Tosic, B. Pagurek, and K. Patel, “WSOL - A language for the formal specification of various constraints and classes of service for web services,” in ICWS, vol. 3. IEEE, 2002, pp. 375–381.

[37] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, “Web service level agreement (WSLA) language specification,” IBM Corporation, 2003. [Online]. Available: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[38] A. Andrieux, K. Czajkowski, A. Dan, and K. Keahey, “Web services agreement specification (WS-Agreement),” Global Grid Forum, Tech. Rep., 2004. [Online]. https://www.ggf.org/Public/ Comment/ Docs/Documents/Oct-2006/WS-AgreementSpecificationDraftFinal/ sp/ tn/ jpver/ v2.pdf

[39] K. T. Kearney, F. Torelli, and C. Kotsokalis, “SLA*: An abstract syntax for Service Level Agreements,” in GRID. IEEE, 2010, pp. 217–224.

[40] M. G. Jaatun, K. Bernsmed, and A. Undheim, “Security SLAs – An Idea Whose Time Has Come?,” in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 123–130. DOI:10.1007/978-3-642-32498-7_10

[41] R. R. Henning, "Security Service Level Agreements: Quantifiable Security for the Enterprise?," in Proceedings of the New Security Paradigms Workshop, 1999.

[42] D. Serrano et al., “SLA guarantees for cloud services,” Future Generation Computer Systems, vol. 54, pp. 233–246, Jan. 2016, DOI: 10.1016/j.future.2015.03.018.

[43] K. A. Torkura, F. Cheng and C. Meinel, "Application of quantitative security metrics in cloud computing," 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 256-262. DOI: 10.1109/ICITST.2015.7412101

[44] OpenStack Cloud Software,
Internet: http://www.openstack.org/ (Access date: 9 December 2018)

[45] Takeshi Takahash, “6-4 Description and Negotiation Techniques to Establish Security SLA”, Journal of the National Institute of Information and Communications Technology Vol. 63 No. 2 (2016), pp 161-165
http://www.nict.go.jp/publication/shuppan/kihou-journal/journal-vol63no2/journal-vol63no2-06-04.pdf

[46] D. Serrano et al., "Towards QoS-Oriented SLA Guarantees for Online Cloud Services," 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, 2013, pp. 50-57. DOI: 10.1109/CCGrid.2013.66

[47] L. Krautsevich and F. Martinelli and A. Yautsiukhin, “Formal approach to security metrics: what does more secure mean for you?” Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pages 162–169, ACM, 2010.

[48] Nia et al. “Enhancing Information Security in Cloud Computing Services using SLA based metrics.” (2011). School of Computing - Blekinge Institute of Technology, 2011.

[49] M. A. Bishop, Computer Security: Art and Science. Addison-Wesley Professional, 2002. https://www.ggf.org/Public/ Comment/ Docs/Documents/Oct-2006/WS-

[50] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds,” Proc. 16th ACM conference on Computer and communications security, 2009, pp. 199-212

[51] Z. Xiao and Y. Xiao, "Security and Privacy in Cloud Computing," in IEEE Communications Surveys & Tutorials, vol. 15, no. 2, pp. 843-859, Second Quarter 2013. DOI: 10.1109/SURV.2012.060912.00182

[52] V. Casola, A. De Benedictis, M. Rak and U. Villano, "SLA-Based Secure Cloud Application Development: The SPECS Framework," 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, 2015, pp. 337-344, DOI: 10.1109/SYNASC.2015.59.

[53] W. A. Ghumman, “Automation of the SLA Life Cycle in Cloud Computing,” in Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 557–562. DOI: 10.1007/978-3-319-06859-6_51

[54] A. Chakrabarty and K. Chuan Tan, “The current state of six sigma application in services,” Managing Service Quality, vol. 17, no. 2, pp. 194–208, Mar. 2007, DOI: 10.1108/09604520710735191.

[55] C. Hoff, P. Simmonds “Security Guide for Critical Areas of Focus in Cloud Computing V.3.0” Cloud Security Alliance Guide. 2011

[56] C. A. da Silva and P. L. de Geus, "An approach to security-SLA in cloud computing environment," 2014 IEEE Latin-America Conference on Communications (LATINCOM), Cartagena de Indias, 2014, pp. 1-6, DOI:10.1109/LATINCOM.2014.7041843.

[57] Microsoft Corporation. Microsoft Windows Azure. Online: http://www.windowsazure.com. Accessed in November 15, 2018

[58] Amazon Web Services, Inc. Amazon CloudWatch. Online: http://aws.amazon.com/cloudwatch. Accessed on November 15, 2018

[59] Google, Inc. Google App Engine. Online: https://developers.google.com/appengine/ Accessed on November 15, 2018

[60] Hyperic. CloudStatus. Online: http://www.hyperic.com/products/cloud-status-monitoring Accessed on November 15, 2018