Asia CCS 2016 http://meeting.xidian.edu.cn/conference/AsiaCCS2016/home.html

Identifying and Utilizing Dependencies Across Cloud Security Services

Ahmed Taha
Technische Universität
Darmstadt, Germany

Patrick Metzler
Technische Universität
Darmstadt, Germany

Ruben Trapero
Technische Universität
Darmstadt, Germany

Jesus Luna
Technische Universität
Darmstadt, Germany

Neeraj Suri
Technische Universität
Darmstadt, Germany

ABSTRACT

Security concerns are often mentioned amongst the reasons why organizations hesitate to adopt Cloud computing. Given that multiple Cloud Service Providers (CSPs) offer similar security services (e.g., "encryption key management") albeit with different capabilities and prices, the customers need to comparatively assess the offered security services in order to select the best CSP matching their security requirements. However, the presence of both explicit and implicit dependencies across security related services add further challenges for Cloud customers to (i) specify their security requirements taking service dependencies into consideration and (ii) to determine which CSP can satisfy these requirements.

We present a framework to address these challenges. For challenge (i), our framework automatically detects conflicts resulting from inconsistent customer requirements. Moreover, our framework provides an explanation for the detected conflicts allowing customers to resolve these conflicts. To tackle challenge (ii), our framework assesses the security level provided by various CSPs and ranks the CSPs according to the desired customer requirements. We demonstrate the framework's effectiveness with real-world CSP case studies derived from the Cloud Security Alliance's Security, Trust and Assurance Registry.

Keywords

Cloud security, security quantification, security service level agreements, service dependencies

1. INTRODUCTION

Cloud computing offers a model where resources (storage, server, etc.) are abstracted and provided "as-a-service" in a remotely accessible manner. In such a service-based environment, service provisioning relies on a service level agreement (SLA) which represents a formal contract established between the Cloud Service Customer (CSC) and the Cloud Service Provider (CSP). The SLA specifies how provisioning takes place as well as the respective rights and duties of the CSC and the CSP. Furthermore, the SLA includes the list of Service Level Objectives (SLOs) which are the measurable elements of an SLA that specify the Cloud services levels requested by the customers, and required to be achieved by the CSP. With security as a major driver, different stakeholders in the Cloud community[1] have identified that specifying security parameters in SLAs (termed as security SLAs or simply secSLAs) is useful to establish common semantics to provide and manage security assurance from two perspectives, namely (i) the security level being offered by a CSP, and (ii) the security level being requested by a Cloud customer.

With the growth of public Cloud security services, multiple CSPs offer "similar" services at different prices and capabilities. However, the offered services are typically bundled together with explicit and implicit dependency[2] relations across them to complicate selection of the single appropriate CSP. For example, the "encryption key management" service in the Cloud depends on several factors such as (a) the techniques used to store the encryption keys, (b) the processes specifying how keys are accessed, (c) the possibility of the key recovery, and finally (d) the control and management of each key. Each of these factors contains different levels of services (e.g., different techniques to store and distribute the keys) which the customer can require and the CSP agrees to fulfill. Most of these factors also depend on each other. These dependency relations increase the difficulties of the customers for finding the single CSP that satisfies their requirements since these relations can easily introduce conflicts; for instance, a customer may require an unachievable level[3] of a dependent security service which cause these requirements to be impossible to satisfy. Moreover, a customer requirement may influence or be influenced by other requirements. Consequently, a CSP being unaware of re-

[1] For example, the European Network and Information Security Agency (ENISA), Cloud Security Alliance (CSA), ISO/IEC, and the European Commission

[2] Dependency relations between services or simply service dependencies are the direct relations between one or more services, where a service can depend on data or resources provided by another service

[3] Unachievable service level exists when a service depends on resources which are not provided by the corresponding dependent service

lated dependencies can erroneously agree on providing an unachievable level of security service according to the customer requirement. Naturally, the CSP will not be able to fulfill this requirement which results in a secSLA violation. As the number of Cloud security services grow, the number of dependencies across the security services also increases making it more likely for customers to introduce conflicts. Also it becomes harder to manually detect and identify the causes of these conflicts especially when multiple types of dependency relations are involved. Therefore, customers need to first consider the dependencies between security services that span across their security requirements specification, and then assess the security services offered by different CSPs to rank different CSPs based on the customer security requirements and priorities.

Although the state of the art predominantly focuses on the methodologies to evaluate and assess Cloud secSLAs [18, 27, 12, 21], most of these methodologies do not account for information about dependencies between services. Overall, it is important to provide customers with comprehensive support that can enable an automatic detection of conflicts and explanations for the dependent relations.

1.1 Contributions

This paper aims to solve the aforementioned issues by proposing a framework for the (i) analysis of the secSLAs service dependencies with the handling of all conflicts, and (ii) the selection of security services. This is done by:

1) Proposing a dependency representation model for validating the secSLAs by checking the existence of conflicts that occur due to different dependency relations between services. The process of analyzing conflicts is both iterative and interactive.

2) Assisting customers who could not resolve the conflicts and thus could not specify their requirements by representing the security requirements in an easy dependent ordered structure using Design Structure Matrix (DSM). In this structure, the security services are ordered according to their level of dependency. This makes the secSLA services and dependencies explicit and traceable regardless of the number of security services.

3) Ranking the CSPs according to the customer requirements in order to find the best matching CSP.

4) Validation of the proposed framework by evaluating CSPs secSLAs found on the public CSA STAR (Security, Trust and Assurance Registry) [8] repository.

To the best of our knowledge, our approach is the first attempt to provide customers with (a) a wide range of support covering conflict detection, (b) outlining dependency conflicts for problematic customers requirements, and (c) a framework to assess and rank the CSPs according to the customer requirements. The rest of the paper is organized as follows. Section 2 develops the background and the basic terminology related to Cloud secSLAs and the service dependencies. The architecture of the proposed framework is elaborated in Section 3. Section 4 presents a real-world use-cases validating the Cloud services evaluation as well as dependency management. Section 5 describes the related work.

2. BASIC CONCEPTS

2.1 Security Service Level Agreements

A Cloud security Service Level Agreement (secSLA) describes the provided security services, and represents the binding commitment between a CSP and a customer. Basically, this outlines the desired security services, each of which contains a list of SLOs. Each SLO is composed of one or more metric values that help in the measurement of the Cloud SLOs by defining parameters and measurement rules that facilitate assessment and decision making. Based on the analysis of the state of practice presented in [14], Cloud secSLAs are typically graphically modeled using a hierarchical structure, as shown in Figure 1. The root of the structure defines the main container for the secSLA. The intermediate levels (second and third levels in Figure 1) are the services which form the main link to the security framework used by the CSP. The lowest level (SLO level) represents the actual SLOs committed by the CSP and consequently offered to the Cloud customer. These SLOs are the threshold values which are specified in terms of security metrics.

It is worth noting that the process of modeling values to a quantitative metric is not straightforward as SLOs can have varied types/ranges of composite qualitative and quantitative values. Hence, we introduce the notion of a "security level" associated to each SLO of the secSLA. To formalize this concept we introduce the following definition.

Definition 1. A secSLA consists of a set of services $S = s_1, ..., s_n$. Each service s consists of finite positive number n of SLOs k_i; where $i = 1 \ldots n$. Each SLO k_i consists of m different metric values v_i; such that $k_i = v_{i,1}, v_{i,2}, ..., v_{i,m}$. Each value implies a different security level offered by the CSP and required by the customer. The total order of security levels k_i is defined using an order relation " $<_i$ "; such that $k_i = v_{i,1} < v_{i,2} < ... < v_{i,m}$. Each k_i value is mapped to a progressive numerical value according to its order. These numerical values are then normalized with respect to the k_i's number of values (m) such that $k_i = \frac{1}{m} < \frac{2}{m} < \ldots < \frac{m}{m}$.

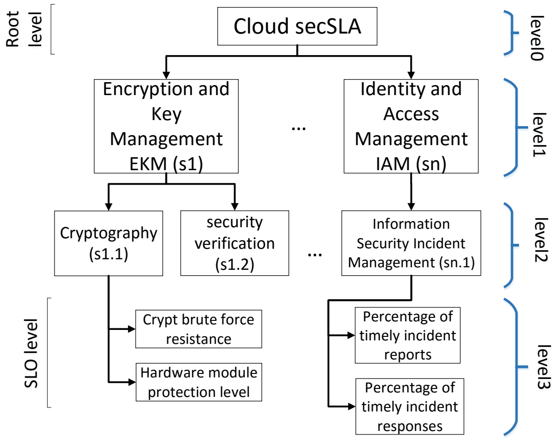

Root level
Cloud secSLA
level0
Encryption and Key Management EKM (s1)
...
Identity and Access Management IAM (sn)
level1
Cryptography (s1.1)
security verification (s1.2)
...
Information Security Incident Management (sn.1)
level2
SLO level
Crypt brute force resistance
Hardware module protection level
Percentage of timely incident reports
Percentage of timely incident responses
level3

Figure 1: Cloud secSLA hierarchy

An example of an SLO, as shown in Figure 1, is "Percentage of timely incident reports" SLO which is composed of

$\{yearly < halfyearly < monthly < weekly\}$ values which are defined using security levels as $level_1 < level_2 < \ldots < level_4$ respectively. These security levels correspond to $\{\frac{1}{4} < \frac{2}{4} < \ldots < \frac{4}{4}\}$. Let us consider a CSP committing "Percentage of timely incident reports" such that the CSP's secSLA specify: Percentage of timely incident report= $level_2$ such that $v_i = \frac{2}{4}$. A CSP commits other SLOs in a similar manner such that the overall CSP's secSLA contains a list of SLOs with different values that the CSP is committed to fulfill. If any of these committed values is not fulfilled by the CSP, then the secSLA is violated.

2.2 Service Dependencies

A service dependency is a directed relation between the services offered in Cloud scenarios. It is expressed as a $1:n$ relationship where one service (termed as dependent) depends on one or multiple services (termed as antecedent). A service can depend on data or resources provided by another service. A service s_1 is dependent on service s_2 if the provisioning of s_1 is conditional to the provisioning of s_2. Explicit knowledge about dependencies is needed to support the management of secSLA by both CSPs and customers. Several types of dependencies are used in literature such as Quality of Service (QoS), price, resource and time dependencies [30]. In this paper we only consider resource dependencies which are validated by matching the security values of the dependent and antecedent SLOs specified in their secSLAs.

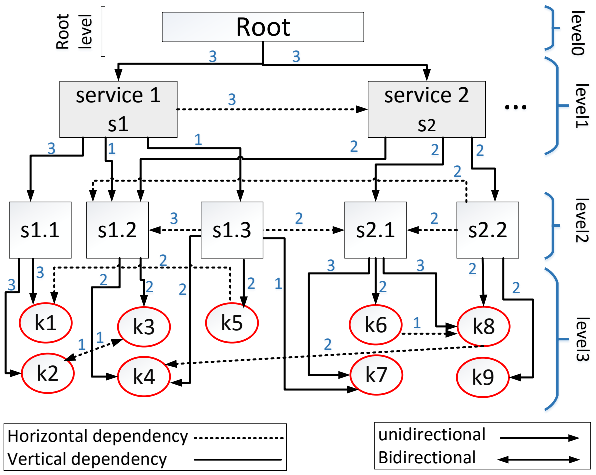


Figure 2: SecSLA hierarchy showing dependencies

We classify dependencies based on their occurrence between services and/or SLOs at the same hierarchical level (horizontal dependencies), as well as between different levels in the hierarchical structure (vertical dependencies) [29] as shown in Figure 2. Dependencies can be further classified into direct and indirect dependencies. Indirect dependencies occur between services which do not directly interact with each other, but where a transitive relationship exists via an intermediate service. In many cases horizontal and vertical dependencies occur at the same time and both dependencies affect the whole composition hierarchy. We also consider different level of dependencies importance presented using a three level scale as shown in Table 1 and Figure 2.

All the dependencies explained so far are considered unidirectional dependencies. Other dependencies as bidirectional (interdependent relations between services) may occur as well. Bidirectional dependency occurs between services s_1 and s_2 if the provisioning of s_1 is conditional to the provisioning of s_2 and at the same time the provisioning of s_2 is conditional to the provisioning of s_1.

We assume that dependencies between services and SLOs in the secSLA are predefined and described by relevant standards working groups. In these groups, the secSLAs contents are defined along with the type of dependencies and associated dependency importance levels. These sets of dependencies are categorized in the secSLA template. This template is later used in the creation of the dependency model and for the SLOs validation (cf., Section 3.2).

Table 1: Dependency importance level

Numeric scale	Meaning
1	Weak dependency
2	Medium dependency
3	Strong dependency

3. PROPOSED FRAMEWORK

We now develop the framework to achieve the quantitative assessment of CSPs security levels where the CSPs are ranked (as per their secSLAs) for the best match to the customer requirements. As an overview of our framework, the dependency management and CSPs ranking are performed in progressive stages as shown in Figure 3.

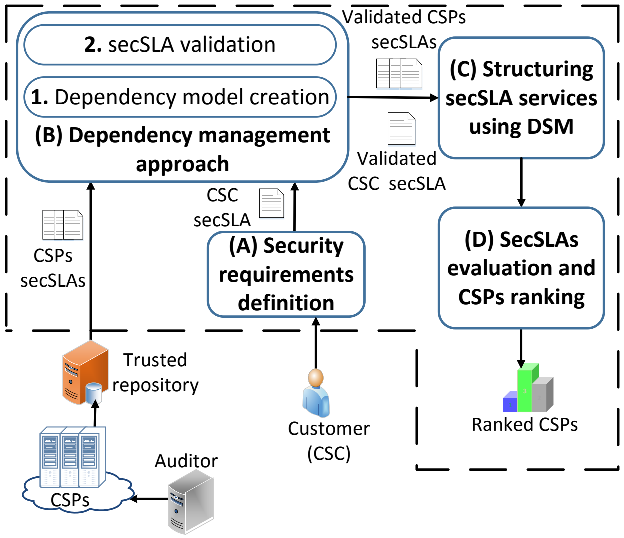

2. secSLA validation
1. Dependency model creation
(B) Dependency management approach
Validated CSPs secSLAs
(C) Structuring secSLA services using DSM
Validated CSC secSLA
CSC secSLA
CSPs secSLAs
(A) Security requirements definition
(D) SecSLAs evaluation and CSPs ranking
Trusted repository
Customer (CSC)
Ranked CSPs
CSPs
Auditor

Figure 3: Proposed methodology stages

After the CSPs submit their secSLAs and the customers specify their security requirements in Stage (A), a dependency model is created in Stage (B) to capture information about secSLAs services and the dependencies that occur between them. This model is specified using a machine readable format to allow automated validation for checking service conflicts and different SLOs compatibility issues. Subsequently in Stage (C), the validated secSLAs are structured using the DSM with the structured secSLA depicting dependencies between services as an ordered listing. The data from the preceding stage feeds into Stage (D) to assess and rank the CSPs according to the customer requirements. In order to guarantee the validity of the proposed framework, the secSLAs provided by the participating CSPs are required to come from a trusted source. In a real-world setup, the trust relationships can be given by an *Auditor* performing a third-party attestation of the CSP secSLA (e.g., through a scheme such as the CSA Open Certification Framework (OCF) [7]). The audited secSLAs are then stored by the CSPs in a trusted repository of SLAs (e.g., the CSA STAR repository [8]) as shown in Figure 3. This trust assumption relies on the fact that the certifications and the repository are trusted, and that the published secSLAs were valid at the time of issuing the corresponding certification or publishing the information in the repository. Stronger assurance levels can be provided by mechanisms such as continuous monitoring (e.g., the CSA STAR continuous monitoring defined at the level 3 of the OCF). The described trust model is able to mitigate the risk of having malicious CSPs publishing false secSLA information with the goal of achieving higher scores in the proposed evaluation system. Other minor risks (e.g., tampered evaluation systems' software) can be mitigated through traditional security controls and secure software development techniques.

3.1 Stage A: Security Requirements Definition

During this stage, the customers create their set of security requirements based on the same secSLA template used by the CSPs to specify their offered security services (as the one shown in Figure 2). The output of this stage is a customer secSLA which is then used, along with one or more CSP secSLAs, as an input to the next stage.

3.2 Stage B: Dependency Management Approach

The approach for managing service dependencies builds on a dependency model, which is used to capture information about security services and the dependencies that occur between them. In order to model service dependencies, it is important to first derive the expected requirements that the dependency model should support.

1. *Support of different dependency types.* The dependency model should support different types of dependencies as well as various dependency classifications (e.g., Horizontal, vertical, unidirectional and bidirectional dependencies).

2. *Support of multiple dependencies.* One security service can have dependencies to several other security services. These dependencies could be of the same or of different types.

3. *Dependency model validation.* It should be possible to automatically validate the dependency model to avoid inconsistencies and conflicts.

This dependency management approach is performed in two phases as shown in Figure 3.

3.2.1 Phase 1: Dependency Model Creation

Handling all the dependencies in the secSLA is a very time consuming and complex task. Therefore, a dependency model is created for each secSLA to cover all identified dependencies within the secSLA. This model is used to capture information about services (each composed of a set of SLOs) and the dependencies that occur between them. We model a secSLA by a tuple $secSLA = (S, l, \rightarrow_S, K, \rightarrow_K, v)$ where:

- S is a set of services s with associated hierarchy levels $l(s) \in \{0, 1, \ldots, n-1\}$. In this paper the secSLA is composed of four hierarchical levels as shown in Figure 2. A *secSLA* contains exactly one service s with $l(s) = 0$, which is the root service. Level n is the SLO level.

- $\rightarrow_S \subseteq S \times (S \cup K) \times \{1, 2, 3\}$ models service dependencies where $\{1, 2, 3\}$ shows the dependency importance level w (cf., Table 1).

- We write $s_1 \xrightarrow{w}_S s_2$ if s_1 (dependent service) depends on s_2 (antecedent service) with dependency importance level w, where $w \in \{1, 2, 3\}$. We write $s \rightarrow_S o$ to express that $s \xrightarrow{w}_S o$ for some $w \in \{1, 2, 3\}$ (where o is either a service or an SLO).

- K is a set of SLOs with associated hierarchy level $l(k) = n$ for all $k \in K$.

- $\rightarrow_K \subseteq K \times K \times \{1, 2, 3\}$ models SLO dependencies. We have $k_1 \xrightarrow{w}_K k_2$ if k_1 (dependent SLO) depends on k_2 (antecedent SLO) with importance level w.

- $v : K \mapsto V$ is an assignment of values in V to SLOs, where V is the set of all metric values of each SLO in K.

- Constraints on the SLO dependency relation are specified using a constraint set $C_v^{\rightarrow_K} \subseteq K \times K \times \{=, \neq, <, \leq, >, \geq\}$. A constraint $(k_1, k_2, \equiv) \in C_v^{\rightarrow_K}$ is satisfied if the values of k_1 and k_2 are related by the given comparison, i.e., $v(k_1) \equiv v(k_2)$. A dependency relation $k_1 \rightarrow_K k_2$ is called valid, written $valid_{C_v^{\rightarrow_K}}(k_1, k_2)$, if the relation satisfies all its constraints, i.e., $\forall (k_1', k_2', \equiv) \in C_v^{\rightarrow_K}.(k_1 = k_1'$ and $k_2 = k_2') \Rightarrow v(k_1) \equiv v(k_2)$.

- We write the transitive closure of $\rightarrow_S$ as $\rightarrow_S^+$, i.e., $s_1 \rightarrow_S^+ s_2$ if $s_1 \rightarrow_S s_2$ or $\exists s_3 \in S.s_1 \rightarrow_S s_3$ and $s_3 \rightarrow_S^+ s_2$. A dependency $o_1 \rightarrow o_2$, where $\rightarrow \in \{\rightarrow_S, \rightarrow_K\}$, between two objects $o_1, o_2 \in S \cup K$ is called symmetric if also $o_2 \rightarrow o_1$. Otherwise, $o_1 \rightarrow o_2$ is called non-symmetric. In other words, o_1 and o_2 are symmetrically dependent if $o_1 \rightarrow o_2$ and $o_2 \rightarrow o_1$.
  Note that it is possible that $o_1 \rightarrow o_2$ is symmetric while the relation $\rightarrow$ is non-symmetric. We explain this using an example, let us consider $\rightarrow = \{(o_1, o_2, w), (o_2, o_1, w), (o_1, o_3, w)\}$ which is a non-symmetric relation and where we have that $o_1 \rightarrow o_2$ is symmetric.

- A secSLA has to satisfy the following constraints:
  i) Services do only depend on services of the same or the next lower hierarchy level: $\forall s_1, s_2 \in S.s_1 \rightarrow_S s_2 \Rightarrow l(s_1) = l(s_2)$ or $l(s_1) + 1 = l(s_2)$
  ii) Only services of hierarchy level $n-1$ depend on SLOs: $\forall s \in S \forall k \in K.s \rightarrow_S k \Rightarrow l(s) = n-1$

iii) Services do not depend on themselves: $\forall s \in S. \neg s \to_S s$

iv) All services depend directly or indirectly on an SLO: $\forall s \in S \exists k \in K. s \to_S^+ k$

3.2.2 *Phase 2: SecSLA Validation*

A meta-model is developed based on the dependency definitions. This meta-model allows the description of SLOs along with information on the secSLA drafted for it. The meta-model is specified using a machine readable format (allowing fully automatic validation) such as an XML data structure using an XML Schema. In this Schema, dependency relations between services are modeled by including the involved service SLOs and their roles as dependent or antecedent as well as their values. Moreover, the constraint comparison is extracted and modeled in the Schema. A brief excerpt from the secSLA dependency model is presented in Appendix A.

Following the development of the dependency model, the secSLA SLOs are validated as depicted in Figure 4. The validation is done by first extracting the secSLA ID, dependency model ID and dependency ID of each two dependent SLOs (each dependency relation in the same dependency model has a unique ID) defined in the XML Schema. Furthermore, for each dependent relation the antecedent and dependent SLO's values are extracted. This entails extracting the constraint comparative (i.e., $=, \neq, \leq, \geq$) and checking if the two dependent SLO values satisfy the constraint. If the constraint between dependent SLOs is not satisfied, the validation scheme shows a conflict between these two SLOs. The dependency ID and dependent SLO ID of the affected SLOs are saved in a list, while the evaluation is continued to determine further conflicts. At the end of this phase a list of all conflicts found in the CSP's secSLA with the conflicts explanation is sent to the CSP in order to resolve these conflicts and resubmit his/her secSLA again to be validated. Similarly, a list of all conflicts found in the customer requirements (specified using the customer secSLA) are resolved by the customer and validated again. If no problems are detected, both the validated CSPs secSLAs along with the customer secSLA are used as an input to Stage (C).

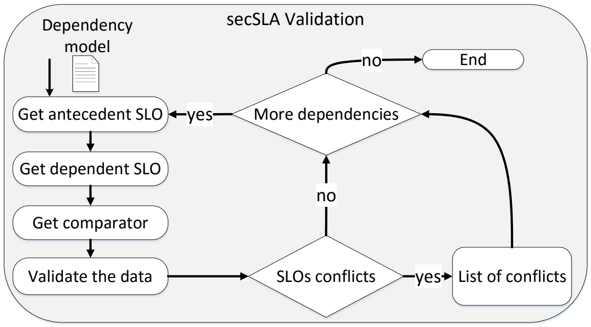


Figure 4: Dependency based secSLA validation stages

3.3 Stage C: Structuring SecSLA Services

In a secSLA, as the number of offered services and SLOs along with the dependencies between them increases, the secSLA hierarchical structure (shown in Figure 2) quickly becomes cluttered and a disorderly network of tangled arcs. This makes it hard for (i) the customers to specify their requirements and resolve the conflicts (which requires an expert customer and is time consuming) and (ii) the CSPs to check the dependency relations between their offered services to avoid any violation. Consequently, the objective of this stage is to embody the secSLA hierarchical structure by mapping the dependencies in a precise order where services and SLOs are ordered according to their level of dependency. This ordering makes the dependency relations explicit and more traceable regardless of the size which allows customers to (a) easily define their security requirements and (b) assess and rank the CSPs according to their security requirements. Furthermore, this provides CSPs with the guidance on the security improvements that should be performed in order to achieve the customer requested security level.

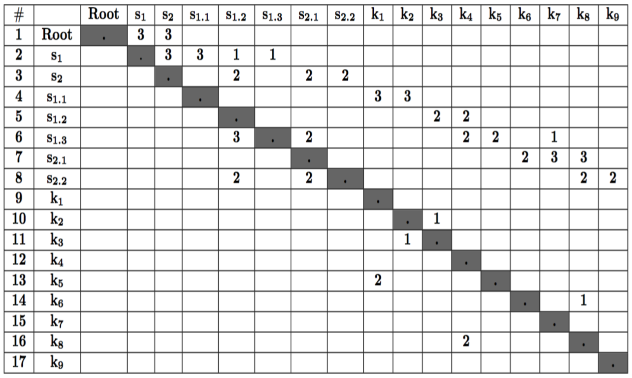
A variety of techniques exist for the analysis, management and ordering of the secSLA services and SLOs other than the graphs used in building the secSLA hierarchical structure. One of these techniques is the program evaluation and review technique [28]. Although this technique incorporates more information than the directed graphs, it is still inadequate for representing the vast majority of design procedures where iteration[4] task relationships are involved. Another technique which has been widely used in documenting design procedures is the structured analysis and design technique [22]. This technique attempts to overcome the size limitations by restricting the amount of information that can be placed on each document. Unfortunately, loops remain an unsolved problem [19].

A representation which overcomes the size and the iteration tasks limitations of those discussed above is the Design Structure Matrix (DSM) (also known as "Dependency Structure Matrix") [26]. There are two main categories of DSMs: static and time-based [2]. Static DSM represents system elements existing simultaneously, such as components of a product architecture or groups in an organization. In time-based DSM, which is the type of DSM used in this paper, the ordering of the rows and columns indicates a flow through time. The DSM embodies the structure of the underlying design activity by mapping the relations between services in a precise order which makes the secSLA clear and easy to read; regardless of the size. To clarify, a secSLA of n services is represented as an $n \times n$ matrix with identical row and column labels. The matrix element a_{ij} is empty if the i^{th} column is independent on the j^{th} row, and not empty if they are dependent. This means, services and SLOs with empty rows have all required information and do not depend on others. Furthermore, the empty columns provide no information required by other services and SLOs.

To demonstrate the idea of DSM, the mapping of the secSLA shown in Figure 2 into a DSM is presented in this section and is depicted in Figure 5. As the dependencies of services on themselves are not considered (as specified in the dependency model constrains in Section 3.2), there are no marks along the diagonal. The strength of the dependencies is given using numerical values; these values provide more detailed information on the relationships between the different system services [26]. In this paper we use the three

[4] A loop of information which occurs if there are bidirectional relations between services, which means each service is waiting for information from the other one

level scale dependency importance rating defined in Table 1. Examining row 2 we note that s_1 strongly depends on s_2 and $s_{1.1}$, and weakly depends on $s_{1.2}$ and $s_{1.3}$. Examining row 10 we note that SLO k_2 weakly depends on k_3.



#		Root	s_1	s_2	$s_{1.1}$	$s_{1.2}$	$s_{1.3}$	$s_{2.1}$	$s_{2.2}$	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9
1	Root	.	3	3														
2	s_1		.	3	3	1	1											
3	s_2			.		2		2	2									
4	$s_{1.1}$				.					3	3							
5	$s_{1.2}$					.						2	2					
6	$s_{1.3}$					3	.	2					2	2		1		
7	$s_{2.1}$							.							2	3	3	
8	$s_{2.2}$					2		2	.								2	2
9	k_1									.								
10	k_2										.	1						
11	k_3										1	.						
12	k_4												.					
13	k_5									2				.				
14	k_6														.		1	
15	k_7															.		
16	k_8												2				.	
17	k_9																	.

Figure 5: DSM mapping of the secSLA shown in Figure 2

After mapping the secSLA into a DSM, we can start reordering the DSM rows and columns in order to transform the DSM into a lower triangular form (that is, the matrix has no entries above the diagonal), this is called DSM partitioning [13] and is done in two steps:

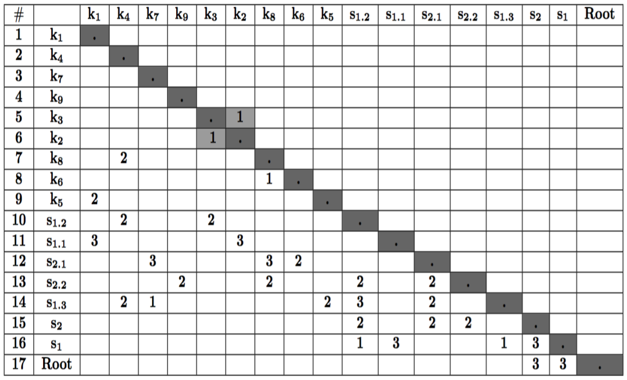
Step 1: Services which have a minimum number of dependencies (initially there will be none) are placed at the top of the DSM. These services are identified as services with minimum number of row values. If there is more than one such service, the one with maximum number of column values is selected.

Step 2: Services that deliver no information to others in the matrix are placed at the bottom of the DSM. This is easily identified by observing an empty column in the DSM. Once a service is rearranged, it is removed from the DSM and step 2 is repeated on the remaining elements.

Figure 6 shows the result of partitioning the DSM depicted in Figure 5. Bidirectional dependencies occur when the matrix cannot be reordered to have all matrix elements subdiagonal. As shown in Figure 6, k_2 and k_3 are bidirectionally dependent (indicated by shading); k_2 needs the information of k_3 and k_3 needs the information of k_2. If k_2 and k_3 are regarded as a single composite service, the cycle can be eliminated [24].
The output of this stage will be (a) one customer secSLA and (b) one or more CSP secSLAs mapped as a DSM where the services and the SLOs are progressively ordered starting from the least dependent.

3.4 Stage D: Evaluation

The quantitative security level assessment of CSPs (for their match to the customer requirements) is the primary objective of the proposed framework developed in this stage. The challenge is not only how to quantify different SLOs in the secSLAs, but also how to aggregate them with a meaningful metric. To solve these issues, we use a ranking mechanism based on the Analytic Hierarchy Process (AHP) [23].



#		k_1	k_4	k_7	k_9	k_3	k_2	k_8	k_6	k_5	$s_{1.2}$	$s_{1.1}$	$s_{2.1}$	$s_{2.2}$	$s_{1.3}$	s_2	s_1	Root
1	k_1	.																
2	k_4		.															
3	k_7			.														
4	k_9				.													
5	k_3					.	1											
6	k_2					1	.											
7	k_8		2					.										
8	k_6							1	.									
9	k_5	2								.								
10	$s_{1.2}$		2			2					.							
11	$s_{1.1}$	3					3					.						
12	$s_{2.1}$			3				3	2				.					
13	$s_{2.2}$				2			2			2		2	.				
14	$s_{1.3}$		2	1						2	3		2		.			
15	s_2										2		2	2		.		
16	s_1										1	3			1	3	.	
17	Root															3	3	.

Figure 6: Final DSM after partitioning and scheduling

AHP is a widely used method for solving problems related to Multi Criteria Decision Making (MCDM) [31]. The advantages of AHP over contemporary multi-criteria methods is its ability to handle composite qualitative and quantitative attributes, along with its ability to identify inconsistencies across requirements [20]. AHP uses a pairwise comparisons for evaluating the alternatives. As an overview of our evaluation framework, the secSLA assessment and the ranking of CSPs are performed in the following progressive phases.

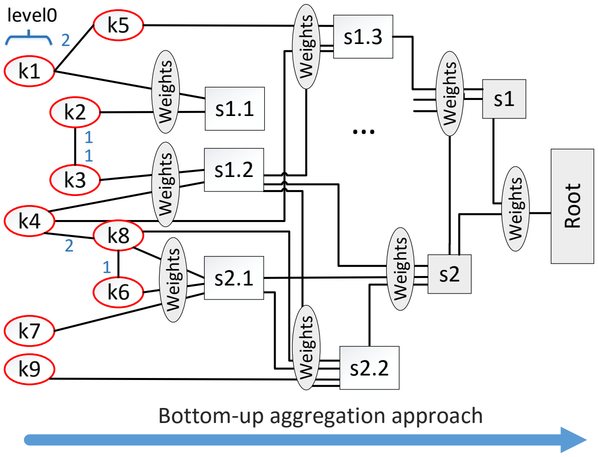


Figure 7: Dependent secSLA hierarchy

Phase 1. Hierarchical Structure

The DSM mapping of each secSLA (either the customer secSLA or the CSPs secSLAs) is modeled as a hierarchical structure in this phase. In this structure the top level of the hierarchical structure defines the main goal and aims to find the overall rank (i.e., the root level). The lowest level is represented by the least dependent SLOs as shown in Figure 7. This hierarchical structure can be used by the basic customers to specify their requirements at the lowest level only (i.e., $level_0$ in Figure 7). We explain the framework presented in this section using a real-world case study in Section 4.
Note that Figure 7 outlines the hierarchical representation

of the DSM shown in Figure 6 where the least dependent SLOs are the SLOs with empty rows in Figure 6.

Phase 2. Weights Assignment

The dependency importance level of each dependency relation (strong, medium or weak dependency with numerical scaling 3, 2 and 1, respectively; cf., Table 1) is specified as a weight (w) during the assessment process. The weight of each relation is used to show how different dependency levels affect the overall security level.

Phase 3. Services Quantification

In order to assess each CSP secSLA, a measurement model for different SLOs should be defined. We use the relative ranking model proposed in [27], which defines the most important requirements used and their quantitative values. The ranking model is based on a pairwise relation of the services (a) provided by different CSPs, and (b) required by customers' such that:

$$CSP_{1,k}/CSP_{2,k} = \frac{v_{1,k}}{v_{2,k}} \tag{1}$$

Where $v_{1,k}$ implies the SLO value provided by $CSP_{1,k}$ (cf., Definition 1) and $CSP_{1,k}/CSP_{2,k}$ indicates the relative rank of $CSP_{1,k}$ over $CSP_{2,k}$ for a particular SLO k. Similarly, $CSP_{1,k}/CSC_k$ indicates the relative rank of $CSP_{1,k}$ over CSC_k, which specifies whether $CSP_{1,k}$ satisfies CSC_k requirements or not. This results in a one to one comparison matrix (CM) of size $(n+1) \times (n+1)$ if there is a total of n CSPs and one CSC for each SLO such that:

$$CM_k =$$

$$\begin{array}{cccc} CSP_{1,k} & \dots & CSP_{n,k} & CSC_k \\ \end{array}$$
$$\left(\begin{array}{cccc} CSP_{1,k}/CSP_{1,k} & \dots & CSP_{1,k}/CSP_{n,k} & CSP_{1,k}/CSC_k \\ \vdots & \ddots & \vdots & \vdots \\ CSP_{n,k}/CSP_{1,k} & \dots & CSP_{n,k}/CSP_{n,k} & CSP_{n,k}/CSC_k \\ CSC_k/CSP_{1,k} & \dots & CSC_k/CSP_{n,k} & CSC_k/CSC_k \end{array}\right) \tag{2}$$

The relative ranking of all the Cloud providers and the customer for each SLO is calculated as a priority vector (PV) of the CM. The PV is an approximation eigenvector of the CM. The PV indicates a numerical ranking of providers that specifies an order of preference among them, as indicated by the ratios of numerical values.

$$\begin{array}{cccccc} & CSP_{1,k} & CSP_{2,k} & \dots & CSP_{n,k} & CSC_k \\ PV_k = & (\; N_{1,k} & N_{2,k} & \dots & N_{n,k} & N_{u,k} \;) \end{array} \tag{3}$$

Where N_1 is a numerical value representing the relative rank of CSP_1 to other CSPs as well as the CSC regarding an SLO. N_u is the relative rank of the CSC required security level with respect to the security levels offered by the CSPs.

Phase 4. Services Aggregation

In the final phase, we follow up with a bottom-up aggregation to give an overall assessment of the security levels and a final ranking of the CSPs. To achieve that, the priority vector of each SLO (Phase 3) is aggregated with their relative normalized weights (dependency importance level) specified in Phase 2. This aggregation process is repeated for all the SLOs in the hierarchy with their relative weights, which results in the ranking of all the Cloud providers based on customer-defined requirements and weights.

$$PV_{aggregated} = \begin{bmatrix} PV_{k_1} & \dots & PV_{k_n} \end{bmatrix} . \begin{bmatrix} W \end{bmatrix}^T \tag{4}$$

Where W is the set of normalized weights of different SLOs such that $W = w_{k_1}, w_{k_2}, \dots, w_{k_n}$. Note that the weights are normalized to satisfy the AHP requirements. PV_{k_1} is the PV calculated for SLO k_1. We demonstrate and validate the framework presented in this section using a real-world case study in Section 4.

4. CASE STUDY: SECURITY EVALUATION

This section shows an empirical validation of the proposed framework through two scenarios that use real world secSLA information derived from the Cloud Security Alliance's STAR repository [8].

4.1 The Customer Perspective: Security Comparison of CSPs

This initial validation scenario demonstrates how a Cloud customer can apply the framework presented in this paper to compare side-by-side three different CSPs based on their advertised secSLAs (compliant with the hierarchy of Figure 2) and with respect to a particular set of security requirements (also expressed as a secSLA). Table 2 presents a sample dataset used for this scenario which is based on the information available in the CSA STAR repository, where the values associated to 15 SLOs for the three selected CSPs are presented. In order to perform a comprehensive validation, the selected SLOs comprised both qualitative and quantitative metrics. The qualitative metrics are specified as security levels (cf., Definition 1) such as *monthly*, *weekly*, and *daily* denoted as security levels $level_1$, $level_2$ and $level_3$, which are modeled as $\frac{1}{3}, \frac{2}{3}, \frac{3}{3}$. Furthermore, no, yes metrics are denoted as $level_0$, $level_1$ respectively. All CSPs security SLOs are normalized to the customer requirements to eliminate masquerading[5]. Furthermore, Table 2 shows two sets of Cloud customer requirements used as a baseline for comparing the selected CSPs:

1) In the Table 2 column marked as Case I, the customer requirements are expressed at a per-SLO granular level. This represents a security-expert customer where the customer specifies his/her requirements at Case I "*req*" column. After that the customer secSLA is validated to check if each constraint between two dependent SLOs is satisfied and at the end of the validation showing the conflicts found. The customer then resolves the SLO conflicts by specifying new values to the SLOs causing conflicts. These new values are shown at Case I "*rev*" column in Table 2.

2) The column marked as Case II shows a set of requirements on SLOs that do not depend on any other SLO (identified using the DSM). These set of SLOs is used to model the customer requirements for the remaining SLOs. This might be the case of a novice or basic customer who can not specify all the secSLA SLOs and resolve the SLO conflicts if any is found.

[5] The masquerading effect happens when the overall aggregated security level values mostly depend on those services with a high-number of SLOs, thus negatively affecting groups with fewer, although possibly more critical, provisions

Table 2: Case Study: Excerpt of secSLA's from CSPs and customer requirements.

Cloud secSLA based on CSA STAR [8]									CSPs			Customer (CSC)		
Services						SLO			CSP_1	CSP_2	CSP_3	Case I		Case II
category	lvl	category	lvl	category	lvl	name	dep.	lvl				req	rev	
Root	3	Audit & Compliance AC	3	Planning $AC1$	2	$AC1.1$	Dep_1		*yes*	*yes*	*yes*	*yes*		*yes*
					3	$AC1.2$	Dep_2	2	$level_2$	$level_2$	$level_3$	$level_3$		
			3	Independent Audits $AC2$	3	$AC2.1$	Dep_1	2	*yes*	*yes*	*yes*	*yes*		
							Dep_3	3						
					3	$AC2.2$	Dep_4	3	*no*	*yes*	*yes*	*no*	*yes*	
					2	$AC2.3$	Dep_4		*no*	*yes*	*yes*	*yes*		
							Dep_5	2						
					1	$AC2.4$	Dep_6	1	*yes*	*no*	*yes*	*yes*		
					1	$AC3.1$			*yes*	*yes*	*yes*	*yes*		
					1	$AC3.2$			*no*	*yes*	*yes*	*yes*		
			3	Regulatory Mapping $AC3$	2	$AC3.1$	Dep_3		*yes*	*yes*	*yes*	*yes*		*yes*
					2	$AC3.2$	Dep_5		*no*	*yes*	*yes*	*yes*		*yes*
					3	$AC3.3$			$level_2$	$level_1$	$level_3$	$level_3$		$level_3$
	3	Business Continuity BC	2	Testing $BC2$	1	$BC2.1$	Dep_7	1	*yes*	*yes*	*yes*	*no*	*yes*	
					1	$BC2.2$	Dep_6		*no*	*yes*	*no*	*no*		*no*
			2	Policy $BC11$	3	$BC11.1$	Dep_7		*yes*	*yes*	*yes*	*yes*		*yes*
					3	$BC11.2$			$level_3$	$level_2$	$level_3$	$level_3$		$level_3$
			2	Regulatory Mapping $AC3$	2	AC3.1			*yes*	*yes*	*yes*	*yes*		
					2	$AC3.2$			*no*	*yes*	*yes*	*yes*		
					3	AC3.3			$level_2$	$level_1$	$level_3$	$level_3$		
	3	Interface Security IS	3	Application Security $IS1$	2	$IS1.1$	Dep_8	1	*weekly*	*weekly*	*daily*	*weekly*	*daily*	*daily*
					1	$IS1.2$	Dep_8	1	$level_2$	$level_2$	$level_3$	$level_3$		$level_3$
							Dep_2							
					1	$AC3.1$			*yes*	*yes*	*yes*	*yes*		

Finally, we consider dependencies between services and SLOs which are going to be validated using the validation model presented in Section 3.2 such that:

SLO dependencies:

- $AC2.1$ is *medium* dependent on $AC1.1$ (this dependency relation is named as Dep_1 in Table 2 and the level of dependency is shown in the SLO "*lvl*" column). This is modeled as $AC2.1 \xrightarrow{2}_K AC1.1$ with constraint $(AC2.1, AC1.1, =) \in C_v^{\rightarrow_K}$.
- $AC1.2$ is *medium* dependent on $IS1.2$ (i.e., named as Dep_2) so that $AC1.2 \xrightarrow{2}_K IS1.2$ with constraint $(AC1.2, IS1.2, =) \in C_v^{\rightarrow_K}$.
- In the same way, Dep_3, Dep_4 and Dep_5 are specified as $AC2.1 \xrightarrow{3}_K AC3.1$ with $(AC2.1, AC3.1, =) \in C_v^{\rightarrow_K}$, $AC2.2 \xrightarrow{3}_K AC2.3$ with $(AC2.2, AC2.3, =) \in C_v^{\rightarrow_K}$ and $AC2.3 \xrightarrow{2}_K AC3.2$ with $(AC2.3, AC3.2, =) \in C_v^{\rightarrow_K}$ respectively.
- Dep_6 is modeled as $AC2.4 \xrightarrow{2}_K BC2.2$ with constraint $(AC2.4, BC2.2, \neq) \in C_v^{\rightarrow_K}$.
- Finally, $IS1.1$ and $IS1.2$ are symmetrically dependent (i.e., Dep_8) so that $IS1.1 \xrightarrow{1}_K IS1.2 \wedge IS1.2 \xrightarrow{1}_K IS1.1$ with constraint $(IS1.1, IS1.2, =) \in C_v^{\rightarrow_K}$.

Service dependencies:

- AC equally depends on $AC1$, $AC2$ and $AC3$ (i.e., equally depends refers to the level of dependency, cf. Table 1, shown in the second column named "*lvl*" in Table 2). So that $AC \xrightarrow{3}_S AC1 \wedge AC \xrightarrow{3}_S AC2 \wedge AC \xrightarrow{3}_S AC3$.
- BC equally depends on $BC2$, $BC11$ and $AC3$. Furthermore, IS *strongly* depends on $IS1$. Each is further depending on the SLOs with different dependency importance levels as shown in Table 2.
- $AC2$ depends on two of $AC3$ SLOs which are $AC3.1$ and $AC3.2$ (shaded in Table 2).
- Since BC depends on $AC3$, and $AC3$ depends on SLOs $AC3.1$, $AC3.2$ and $AC3.3$. Then using transitive closure $BC \rightarrow_S^+ AC3.1 \wedge BC \rightarrow_S^+ AC3.2 \wedge BC \rightarrow_S^+ AC3.3$ (shaded in Table 2).

4.2 The Customer Perspective: Security Comparison of CSPs

4.2.1 Cloud Customer Case I Requirements

In this case, both the customer requirements and the CSPs secSLAs are validated to check conflicts between SLOs based on the defined dependencies as specified in Section 3.2. Errors in the customer requirements are automatically detected based on the modeled dependencies, such that:

- Dep_1 **Validation**: $AC2.1$ security level ($level_1$) is *equal* to the $AC1.1$ security level ($level_1$),$v(AC2.1) = v(AC1.1)$. **Result:** Valid. Furthermore, Dep_2 **Validation**: $AC1.2$ security level ($level_3$) is *equal* to $IS1.2$ security level. **Result:** Valid. Similarly, Dep_3 and Dep_5 are valid.

- *Dep_4* **Validation**: $AC2.2$ security level ($level_0$) is *not-equal* to the $AC2.3$ security level ($level_1$). **Result:** An SLO conflict occurs, thus the customer modifies $AC2.2$ (dependent SLO) to *yes* ($level_1$) to satisfy the dependency constraint (as shown in Case I "*rev*" column in Table 2). In the same way, Dep_7 is validated.

- *Dep_6* **Validation**: Since the $(AC2.4, BC2.2, \neq) \in C_v^{\overrightarrow{K}}$ and $AC2.4$ security level ($level_1$) is *not-equal* to the $BC2.2$ security level ($level_0$), $v(AC2.4) \neq v(BC2.2)$ the constraint is satisfied. **Result:** Valid.

- *Dep_8* **Validation**: $IS1.1$ security level (*weekly* which is $level_2$) is *not-equal* to the $IS1.2$ security level ($level_3$). **Result:** SLO conflict occurs. The customer changes $IS1.1$ to *daily* ($level_3$).

After the customer has resolved all the SLO conflicts and the CSPs secSLAs are validated, each secSLA is mapped to a DSM (cf., Section 3.3) to embody the secSLA hierarchical structure. This structure is used in the ranking of CSPs according to the customer requirements (cf., Section 3.4). The ranking computation process for Cloud security SLOs defined in Table 2 is explained step-by-step, in the rest of this section.
For the Audit & Compliance control of Cloud secSLA, there are three security controls ($AC1, AC2$ and $AC3$) which are further divided to SLOs ($AC1.1$, $AC1.2$, $AC2.1$, $\ldots$). For $AC1.2$ the providers and the customer can specify their SLOs values from $level_1$ to $level_3$. Using the data shown in Table 2, Equation 1 is used to define the $AC1.2$ pairwise relation such that:

$$\begin{aligned} CSP_{1,AC1.2}/CSP_{3,AC1.2} &= \frac{2}{3}/\frac{3}{3} \\ CSC_{AC1.2}/CSP_{2,AC1.2} &= \frac{3}{3}/\frac{2}{3} \end{aligned}$$

Thus, the CM of $AC1.2$ is calculated using Equation 2 as:

$$CM_{AC1.2} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 1 & 1 & 2/3 & 2/3 \\ 1 & 1 & 2/3 & 2/3 \\ 3/2 & 3/2 & 1 & 1 \\ 3/2 & 3/2 & 1 & 1 \end{pmatrix} \end{array}$$

The relative ranking of the CSPs for $AC1.2$ is given by the priority vector of $CM_{AC1.2}$ ($PV_{AC1.2}$) which is calculated using Equation 3.

$$PV_{AC1.2} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.2 & 0.2 & 0.3 & 0.3 \end{pmatrix} \end{array}$$

This implies that only CSP_3 satisfies the customer requirement for $AC1.2$. In a similar way, we calculate $CM_{AC1.1}$ and $PV_{AC1.1}$. The $AC1$ priority vector is then premeditated by aggregating $PV_{AC1.1}$ and $PV_{AC1.2}$ with the normalized dependency levels (which are defined as weights w_{AC1} as specified in Section 3.4) where AC_1 is *medium* dependent on *AC1.1* and *strongly* dependent on *AC1.2* then after normalization:

$$w_{AC1} = \begin{array}{c} \begin{array}{cc} AC1.1 & AC1.2 \end{array} \\ \begin{pmatrix} \frac{2}{5} & \frac{3}{5} \end{pmatrix} \end{array}$$

Thus, PV_{AC1} is calculated using Equation 4 such that:

$$PV_{AC1} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{cc} PV_{AC1.1} & PV_{AC1.2} \end{array} \\ \begin{pmatrix} 0.25 & 0.2 \\ 0.25 & 0.2 \\ 0.25 & 0.3 \\ 0.25 & 0.3 \end{pmatrix} \end{array} \cdot \begin{array}{c} w_{AC1} \\ \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} \end{array}$$

Therefore,

$$PV_{AC1} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.22 & 0.22 & 0.28 & 0.28 \end{pmatrix} \end{array}$$

This means that only CSP_3 satisfies the customer requirements for $AC1$ as shown in Figure 8. Similarly, the Independent Audits and Regulatory Mapping priority vectors are calculated. Subsequently, the three Audit & Compliance services $AC1$, $AC2$, $AC3$ priority vectors are aggregated to have the overall Audit & Compliance priority vector PV_{AC} as:

$$PV_{AC} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{ccc} PV_{AC1} & PV_{AC2} & PV_{AC3} \end{array} \\ \begin{pmatrix} 0.22 & 0.1212 & 0.1667 \\ 0.22 & 0.2727 & 0.2143 \\ 0.28 & 0.3030 & 0.3095 \\ 0.28 & 0.3030 & 0.3095 \end{pmatrix} \end{array} \begin{array}{c} w_{AC} \\ \begin{pmatrix} 0.3333 \\ 0.3333 \\ 0.3333 \end{pmatrix} \end{array}$$

In a similar way, the Business Continuity and Interface Security priority vectors are considered, such that the $IS1$ priority vector is calculated by aggregating $PV_{IS1.1}$, $PV_{IS1.2}$ and $PV_{AC3.1}$ with the normalized dependency levels (w_{IS1}) using Equation 4, as:

$$PV_{IS1} = \begin{array}{c} \begin{array}{ccc} PV_{IS1.1} & PV_{IS1.2} & PV_{AC3.1} \end{array} \\ \begin{pmatrix} 0.2 & 0.2 & 0.25 \\ 0.2 & 0.2 & 0.25 \\ 0.3 & 0.3 & 0.25 \\ 0.3 & 0.3 & 0.25 \end{pmatrix} \end{array} \begin{array}{c} w_{IS1} \\ \begin{pmatrix} 0.5 \\ 0.25 \\ 0.25 \end{pmatrix} \end{array}$$

Therefore,

$$PV_{IS1} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.21 & 0.21 & 0.29 & 0.29 \end{pmatrix} \end{array}$$

This means only CSP_3 satisfies $IS1$ customer requirement as shown in Figure 8. Finally, the priority vectors of Audit & Compliance, Business Continuity and Interface Security are aggregated to obtain the total secSLA priority vector PV_{Root}.

$$PV_{Root} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{ccc} PV_{AC} & PV_{BC} & PV_{IS} \end{array} \\ \begin{pmatrix} 0.1693 & 0.2260 & 0.21 \\ 0.2357 & 0.2267 & 0.21 \\ 0.2975 & 0.2736 & 0.29 \\ 0.2975 & 0.2736 & 0.29 \end{pmatrix} \end{array} \begin{pmatrix} 0.3333 \\ 0.3333 \\ 0.3333 \end{pmatrix}$$

$$PV_{Root} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.2018 & 0.2241 & 0.2870 & 0.2870 \end{pmatrix} \end{array}$$

Consequently, CSP_3 is the only provider that fulfills the customer's requirements. That was expected, as CSP_1 is not offering $AC2.2$, $AC2.3$, $AC3.2$ and is under-provisioning $IS1.1$ and $IS1.2$. CSP_2 is not providing $BC2.2$ and is not fulfilling customer requirements for $AC1.2$, $AC3.3$, $BC11.2$, $IS1.1$ and $IS1.2$. Only CSP_3 fulfills all the customer's requirements. As a result, CSP_3 is the best matching provider according to the customer's requirements, followed by CSP_2 and CSP_1.

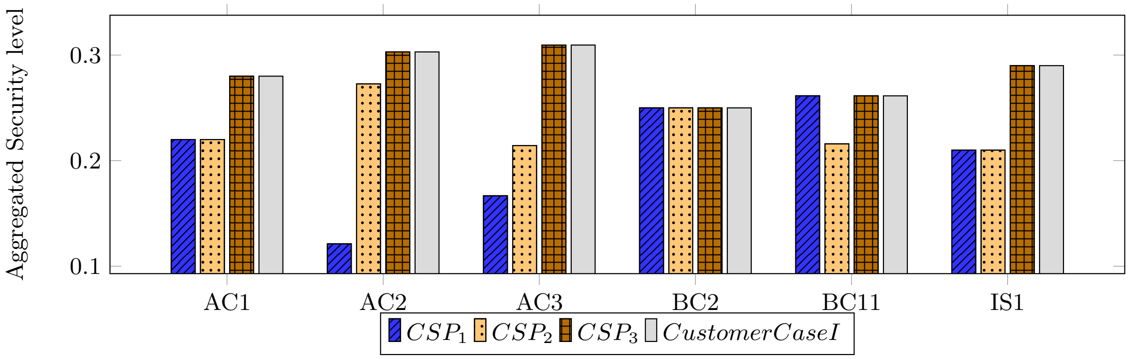


Figure 8: CSPs comparison with respect to Customer Case I requirements.

4.2.2 *Cloud Customer Case II Requirements*

In this case we consider a novice customer who cannot specify his/her precise SLO requirements and/or resolve the SLOs conflicts. The structured secSLA presented using DSM enables the customer to easily specify his/her requirements, regardless of the size of the secSLA and the number of dependencies. In this case the customer defines the least dependent SLOs - these are specified using DSM and are shown in the column marked as Case II in Table 2.

Using the data shown in Table 2, Equation 1 defines the $AC1.1$ pairwise relation as in Case I. Then the relative ranking of the CSPs for $AC1.1$ is given by the priority vector calculated using Equation 3 (as explained in Case I).

$$PV_{AC1.1} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix} \end{array}$$

Similarly, all the lowest level SLOs (least dependent SLOs) are calculated. The $PV_{AC3.1}$, $PV_{AC3.2}$, $PV_{AC3.3}$, $PV_{BC2.2}$, $PV_{BC11.1}$, $PV_{BC11.2}$, $PV_{IS1.1}$ and $PV_{IS1.2}$ are calculated in a similar way. Then based on the DSM order $AC1.2$ is calculated. $AC1.2$ is depending on $IS1.2$ (Dep_1), thus the $PV_{AC1.2}$ is equal to $PV_{IS1.2}$.

$$PV_{AC1.2} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.2 & 0.2 & 0.3 & 0.3 \end{pmatrix} \end{array}$$

In the same way, $PV_{AC2.3}$, $PV_{AC2.4}$ and $PV_{BC2.1}$ are calculated (they are equal to $PV_{AC3.2}$, $PV_{BC2.2}$ and $PV_{BC11.1}$ respectively). Furthermore, $AC2.1$ is calculated. $AC2.1$ depends on $AC1.1$ and $AC3.1$ (Dep_1 and Dep_3) with different levels of dependencies. Thus, using Equation 4:

$$PV_{AC2.1} = \begin{array}{c} \\ CSP_1 \\ CSP_2 \\ CSP_3 \\ CSC \end{array} \begin{array}{c} \begin{array}{cc} PV_{AC1.1} & PV_{AC3.1} \end{array} \\ \begin{pmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \\ 0.25 & 0.25 \\ 0.25 & 0.25 \end{pmatrix} \end{array} \begin{array}{c} w_{AC2.1} \\ \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} \end{array}$$

After all the SLO priority vectors are determined, the priority vectors are aggregated with the dependency importance level to get the overall rank of CSPs according to the customer requirements as specified in Case I. As a result the root priority vector is equal to:

$$PV_{Root} = \begin{array}{c} \begin{array}{cccc} CSP_1 & CSP_2 & CSP_3 & CSC \end{array} \\ \begin{pmatrix} 0.2018 & 0.2241 & 0.2870 & 0.2870 \end{pmatrix} \end{array}$$

4.3 The CSP Perspective: Maximising Offered Security Levels

The second validation scenario presented in this paper applies the secSLA evaluation techniques to solve problems faced by CSPs i.e., which specific security SLO from the offered secSLA should be improved in order to maximise the overall security level according to the customer requirements? This might be the case of a well-established CSP deciding where to invest in order to achieve the highest possible security level, or a new CSP designing the secSLA. To answer this question, we could perform a sensitivity analysis to ascertain the security benefits of improving one or more SLOs. However, this analysis becomes impractical as the number of SLOs and the dependencies between them increase. Thus the sensitivity analysis is performed on the least dependent SLOs identified by the DSM.

We used the CSP_1 dataset described at Table 2, and applied the Case II requirements to setup the customer's baseline for the security evaluation. From the existing 9 least dependent SLOs (Case II column in Table 2) the CSP_1 is under-provisioning 4 of them ($AC3.2$, $AC3.3$, $IS1.1$ and $IS1.2$). Figure 9 shows how the proposed framework can be used to analyse an existing secSLA, and extract the individual SLOs that, if enhanced, would result on different improvements associated to the overall security level. In this case, the X-axis represents the improvement associated to the overall security level after enhancing any of the SLOs. It is shown as a percentage where 0% corresponds to the original secSLA and 100% is the most effective SLO. For example, providing tenants with the security policies applicable to virtualised resources ($AC3.2$ in Figure 9), quantitatively increases CSP_1 security level better than improving the thresholds committed for any of the other SLOs.

5. RELATED WORK

With the rapid growth of the Cloud services, multiple approaches are emerging to assess the functionality and security of CSPs. In [16], the authors proposed a framework to compare different Cloud providers across performance indicators. In [12], an AHP-based ranking technique that utilizes performance data to measure various QoS attributes and evaluates the relative ranking of CSPs was proposed. In [25], a framework of critical characteristics and measures that enable a comparison of Cloud services is presented. However, these studies focused on assessing performance of Cloud services but not their security properties.

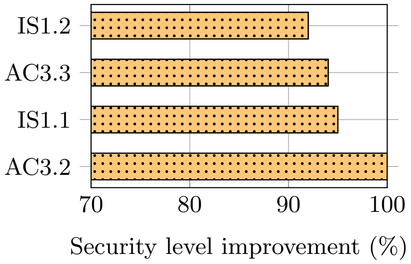


Figure 9: Sensitivity analysis: CSP_1 SLOs that maximise the overall security level.

Security requirements for non-Cloud scenarios have been addressed by Casola et al. [3], who proposed a methodology to evaluate security SLAs for web services. Chaves et al. [5] explored security in SLAs by proposing a monitoring and controlling architecture for web services. In [11] and [15], the authors propose a technique to aggregate security metrics from web services. Their approach focused on the process of selecting the optimal service composition based on a set of predefined requirements. However, the authors did not propose any techniques to assess Cloud secSLAs or empirically validate the proposed metrics.

In [1], the authors propose the notion of evaluating Cloud secSLAs by introducing a metric to benchmark the security of a CSP based on categories. However, the resulting security categorization is purely qualitative and lacks the support of dependencies. Luna et al. [18] presented a methodology to quantitatively benchmark Cloud security with respect to customer defined requirements (based on control frameworks). In [27], the authors presented a framework to compare, benchmark and rank the security level provided by two or more CSPs. However in both of them, the dependencies and conflict detection are not covered.

There has been considerable effort on the conflict analysis of network system management policies. Charalambides et al. [4] expressed QoS policies using Event Calculus for managing DiffServ networks, and their conflict analysis is conducted in a pairwise comparison fashion. Dunlop et al. [9] proposed a model to specify policies of permission, prohibition and obligation in a temporal logic language that can reason about the sequences of events. In [6], the authors presented a framework for automatic detection of conflicts covering violation of enterprise policies and inconsistency of customer requirements. Ensel and Keller [10] introduced an approach to handle dependencies between managed resources (e.g., web application server, database) in a distributed system. However, the support for secSLA management is not provided. The COSMA approach [17] supports the providers of composite services to manage their SLAs. However, COSMA does not support the determination of the effect of SLO violations on other services based on dependency information.

6. CONCLUSIONS

Choosing a Cloud provider that satisfies the security requirements of the customer has become challenging. Quantification and evaluation offer powerful tools for choosing between different CSPs. While the initial results of such techniques are promising, they still lack tackling the dependency relations that span across customer requirements. Most of these methodologies do not account for information about dependencies between services. It is important to provide customers with comprehensive support which enables automatic conflict detection and explanation dedicated to the dependent relations. Our framework automatically detects any conflicts caused by inconsistent customer requirements. Additionally, our framework ranks CSPs and selects the CSP that best satisfies the customer requirements. Explanations of the detected conflicts are generated to identify problematic customer requirements. Using our framework, we evaluated different CSPs based on varied security specifications with respect to the customer security requirements. Additionally, we also addressed different assignments of security levels and weights enabling customers to compare the security levels offered by different CSPs.

Our case study based evaluation showed that our framework effectively validated complicated requirements from different customers and selected the best matching CSP from the set of all CSPs. Currently, we are enhancing our input model of Cloud services by encoding more services from the STAR repository [8].

7. ACKNOWLEDGMENTS

Research supported, in part, by H2020- 644579 (ESCUDO-CLOUD), FP7-ICT-2013-11610795 (SPECS) and DFG SFB CROSSING.

8. REFERENCES

[1] M. Almorsy, J. Grundy, and A. Ibrahim. Collaboration-based cloud computing security management framework. *Proc. of Cloud Computing*, pages 364–371, 2011.

[2] T. Browning. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *In Trans. on Engg. Management*, 48(3):292–306, 2001.

[3] V. Casola, A. Mazzeo, N. Mazzocca, and M. Rak. A sla evaluation methodology in service oriented architectures. *In Quality of Protection*, pages 119–130, 2006.

[4] M. Charalambides, P. Flegkas, G. Pavlou, J. Rubio-Loyola, A. Bandara, E. Lupu, A. Russo, N. Dulay, and M. Sloman. Policy conflict analysis for diffserv quality of service management. *In Network and Service Management*, 6(1):15–30, 2009.

[5] S. Chaves, C. Westphall, and F. Lamin. SLA perspective in security management for cloud computing. *Proc. of Networking and Services*, pages 212–217, 2010.

[6] C. Chen, S. Yan, G. Zhao, B. Lee, and S. Singhal. A systematic framework enabling automatic conflict detection and explanation in cloud service selection for enterprises. *Proc. of Cloud Computing*, pages 883–890, 2012.

[7] Cloud Security Alliance. The Open Certification Framework. *https://cloudsecurityalliance.org/research/ocf/.*

[8] Cloud Security Alliance. The Security, Trust & Assurance Registry (STAR). *https://cloudsecurityalliance.org/star/.*

[9] N. Dunlop, J. Indulska, and K. Raymond. Dynamic conflict detection in policy-based management systems. *Proc. of the Enterprise Distributed Object Computing Conference*, pages 15–26, 2002.

[10] C. Ensel and A. Keller. Managing application service dependencies with xml and the resource description framework. *Proc. of the Integrated Network Management Proceedings*, pages 661–674, 2001.

[11] G. Frankova and A. Yautsiukhin. Service and protection level agreements for business processes. *Proc. of European Young Researchers Workshop on Service Oriented Computing*, pages 38–43, 2007.

[12] K. Garg, S. Versteeg, and R. Buyya. A framework for ranking of cloud computing services. *In Future Generation Computer Systems*, 29(4):1012–1023, 2013.

[13] D. Gebala and S. Eppinger. Methods for analyzing design procedures. *Proc. of Design Theory and Methodology*, pages 227–233, 1991.

[14] J. Luna, A. Taha, R. Trapero, and N. Suri. Quantitative reasoning about cloud security using service level agreements. *In Trans. on Cloud Computing*, (99), 2015.

[15] L. Krautsevich, F. Martinelli, and A. Yautsiukhin. A general method for assessment of security in complex services. *Proc. of Towards a Service-Based Internet*, pages 153–164, 2011.

[16] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. *Proc. of Internet Measurement*, pages 1–14, 2010.

[17] A. Ludwig and B. Franczyk. Cosma–an approach for managing slas in composite services. *Proc. of Service-Oriented Computing*, pages 626–632, 2008.

[18] J. Luna, R. Langenberg, and N. Suri. Benchmarking Cloud Security Level Agreements Using Quantitative Policy Trees. *Proc. of Cloud Computing Security Workshop*, pages 103–112, 2012.

[19] D. Marca and C. McGowan. Sadt: structured analysis and design technique. *McGraw-Hill*, 1987.

[20] R. Ramanathan. A note on the use of the analytic hierarchy process for environmental impact assessment. *In Journal of Environmental Management*, 63(1):27–35, 2001.

[21] Z. Rehman, F. Hussain, and O. Hussain. Towards multi-criteria cloud service selection. *Proc. of Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 44–48, 2011.

[22] D. Ross. Structured analysis (SA): A language for communicating ideas. *In Software Engineering*, (1):16–34, 1977.

[23] T. Saaty. How to make a decision: the analytic hierarchy process. *In European journal of operational research*, 48(1):9–26, 1990.

[24] N. Sangal, E. Jordan, V. Sinha, and D. Jackson. Using dependency models to manage complex software architecture. *In Sigplan Notices*, 40(10):167–176, 2005.

[25] J. Siegel and J. Perdue. Cloud services measures for global use: the service measurement index (smi). *Proc. of Global Conference*, pages 411–415, 2012.

[26] D. Steward. The design structure system: a method for managing the design of complex systems. *In Trans. on Engg. Management*, (3):71–74, 1981.

[27] A. Taha, R. Trapero, J. Luna, and N. Suri. AHP-Based Quantitative Approach for Assessing and Comparing Cloud Security. *Proc. of Trust, Security and Privacy in Computing and Communications*, pages 284–291, 2014.

[28] J. Wiest and F. Levy. A management guide to PERT/CPM. *Prentice-Hall*, 1977.

[29] M. Winkler and A. Schill. Towards dependency management in service compositions. *Proc. of e-Business*, pages 79–84, 2009.

[30] M. Winkler, T. Springer, and A. Schill. Automating composite sla management tasks by exploiting service dependency information. *Proc. of Web Services*, pages 59–66, 2010.

[31] M. Zeleny. *Multiple Criteria Decision Making.* McGraw Hill, 1982.

APPENDIX

A. EXCERPT OF A SECSLA DEPENDENCY MODEL

The extract of a CSP secSLA dependency model with two SLOs (named “User authentication and identity assurance level” (k_{Usauth}) and “CSP-Authentication” (k_{CSauth})) and the dependency relation between them is shown in Listing 1. In the Listing, $k_{Usauth} \rightarrow_K k_{CSauth}$ and the two SLOs security levels are modeled as $v(k_{Usauth})$ and $v(k_{CSauth})$, respectively. The requirement is that the security level of k_{CSauth} is higher than or equal the security level of k_{Usauth}, i.e., $v(k_{CSauth}) \geq v(k_{Usauth})$. This requirement is modeled as $(k_{Usauth}, k_{CSauth}, \leq) \in C_v^{\rightarrow_K}$.
Note that all service levels (e.g., $level_2$, $level_3$, $monthly$, $\ldots$) are modeled as numerical values. These numerical values are the security SLOs values in the XML schema shown in Listing 1.

Listing 1: Excerpt of dependency model of a secSLA

```
<secSLA slaid=‘‘sla1”>
<dependencyModel Depmodelid=‘‘depmod_1”>
    <sloDependency Depid=‘‘dep−1” type=‘‘unidirectional”>
        <dependent depSLOid=‘‘AA1.1_sla1” value =‘‘2”/>
        <antecedent antSLOid=‘‘AA1.2_sla1” value=‘‘3”/>
        <constraint> leq </constraint>
    </sloDependency>
</dependencyModel>
</secSLA>
```