Towards a secure database integration using SLA in a multi-cloud context

Nadia Bennani*, Chirine Ghedira-Guegan†, Genoveva Vargas-Solar‡ and Martin A. Musicante§

*U. de Lyon, CNRS INSA-Lyon, LIRIS, UMR5205, F-69621, France (nadia.bennani@insa-lyon.fr)
†MAGELLAN, IAE, Univ. J-Moulin, Lyon 3, France (chirine.ghedira-guegan@univ-lyon3.fr)
‡ U. Grenoble Alpes, CNRS, LIG-LAFMIA, St. Martin D'Hères, France (genoveva.vargas@imag.fr)
§DIMAp, UFRN, Natal, Brazil (mam@dimap.ufrn.br)

***Abstract*—The recent emergence of the cloud opens new challenges for data processing and integration. Indeed, unlimited access to cloud resources and the "pay as U go" model change the hypothesis for integrating and processing big data collections. This is accentuated due to cloud provider incentives to attract data providers and let them choose their cloud. In this context data services are deployed on several clouds which make data integration face several heterogeneity problems. This paper proposes a secure-aware data integration (lookup, aggregation, correlation) strategies that can be performed on a multi-cloud environment guided by their respective SLA contracts. The resulting data integration is provided as DaaS (data as a service) with a global SLA that consolidates as much as possible the SLA adopted by every cloud provider. Our data integration approach is being tested for integrating databases used for managing an education program.**

***Keywords*-security services, SLA; Cloud Computing; Data Integration;**

I. Introduction

The advent of cloud computing has imposed a new resource consuming model that focuses on the *technical and economic* conditions to be fulfilled in order to access potentially unlimited resources. Integrating and processing heterogeneous data collections, calls for efficient methods for correlating, associating, and filtering them considering their variety (i.e., different formats and data models) and quality, e.g., trust, freshness, provenance, partial or total consistency. Existing data integration techniques have to be revisited considering weakly curated and modeled data sets. This can be done according to (i) quality of service (QoS) requirements expressed by their consumers and Service Level Agreement (SLA) contracts exported by data services; (ii) cloud providers that host these collections and deliver resources for executing data processing and integration processes. In fact, the interest of SLA has been demonstrated in data analysis but has not been yet widely considered for integrating data. We believe in the benefits of SLA-based data integration as an approach for better meeting user requirements related to the conditions in which data is delivered and integrated, and on the quality of the data provided by services. To do so, several granularities of SLA must be considered: first, at the cloud level: the SLA ensured by providers regarding data; then at the service level, as unit for accessing and processing data, to be sure to fulfill specific needs; and finally at the integration level i.e the possibility to process, correlate and integrate big data collections distributed across different cloud storage supports, providing different quality properties for data (trust, privacy, reliability, etc).

To better understand our problem, consider a massive open online course system (MOOC) where users produce and consume content according to the geographic area and the expertise of participants. Producers are characterized according to their location, the type and topic of content they can provide, the access conditions (e.g. cost, inscription, or exchange unit), and the time window in which they can produce content. Consumers are characterized by their location, their interests during a certain interval of time, the maximum total cost they are ready to pay consuming content, or the resources they are ready to provide in order to get the service, and QoS requirements such as availability and how critical it is to consume a given type of content. Both producers and consumers have subscriptions to different cloud providers for dealing with content storage, processing and exchange. The MOOC aims at being privacy respectful of the producers and consumers participating in courses. Providers and consumers can ask to minimize the transfer of personal data when they share/consume content. According to the level of trust associated to data providers, the MOOC can use privacy preserving techniques to let users share content anonymously (Constraint1). Furthermore, data providers can also wish to give restricted data access credentials w.r.t. to their trust level, when their data are used within an integration process (Constraint 2).

From this scenario we identify a set of issues that have to be addressed. First, how can the user efficiently obtain results for her queries such that they meet her QoS requirements while respecting her subscribed contracts with the involved cloud provider(s)? The user knows the low-level clauses of her SLA contract but she does not have an idea about the resources required to satisfy her requirements (SLA exported by services, example of Constraint 1). Second, data services contracts should not be neglected in an integration process (Constraint 2). As data services are deployed in a multi-cloud context, there is a need to

establish the rules to determine how integration can be done and in which conditions. Intuitively, integration can be done enforcing all specified conditions; but it is also possible to expect situations where conditions can only be verified partially. Furthermore, matching data providers with requests and QoS preferences with SLA's can be computationally costly. For this reason the results of such a complex process should be capitalized for further integration requests. How can this be done?

The first step is to propose an architecture that integrates all services participating in a data integration process that copes QoS and SLA's of the services that participate in this process for computing a given query. We have proposed such architecture in [1]. Based on this architecture it is necessary to revisit the integration phases specification to include QoS and SLA. The objective being to propose a new way of dealing with a data production and consumption process, considering SLA formalism and clauses -especially security concerns- and heterogeneity among clouds. Therefore, this paper introduces a preliminary solution addressing SLA-guided data integration focusing on query rewriting addressing security issues. We assume that data services publish Agreed-SLA that describe their static and dynamic QoS measures in specific registries. Thanks to an extended query rewriting process, a new kind of SLA is derived to match user preferences with low level agreed SLAs exported by services. The derived SLA allows to choose services (w.r.t user requirements) that can contribute for answering a query. Once data services have been chosen, the integration process will be guided by an integration SLA, negotiated according to user requirements and inter-cloud contracts.

This paper is organized as follows. Section II presents related works that address SLA modeling, integration and SLA guided data management processes in cloud environments. Section III gives an overview of our approach and highlights the need of new kind of SLA to conduct the integration process. Section IV, V and VI give an idea of the main steps of our proposal. Finally section VII concludes the paper and discusses future work.

II. Related Work

The use of cloud SLA rely on two principles: (i) negotiation of conditions in which services will be provided and that are statically agreed between the parts and (ii) monitoring of these conditions during the use of cloud resources in order to detect SLA contract violation.

SLA are normally expressed in terms of *technical*, quantifiable measures, like the storage space size, the degree of virtualization, the number of requests for a given service. From the user point of view, the interest is normally on *service* delivery measures that are related with her QoS requirements like service availability, resilience, recovery time or cost. The challenge is to couple technical and service delivery measures expressed in SLA in order to agree on the conditions in which they will be considered for executing tasks for a user. Existing works use matching and negotiation techniques for addressing this challenge. For instance, [2] proposes a bottom-up approach based on a cloud component that monitors technical measures, analyses and matches them to service delivery (high level) clauses expressed by the user. [3] describes a *Semantic SLA* that can be understood by all parties including providers, consumers, and monitoring services. Their goal is, starting from a high level SLA, to measure and identify how to derive SLA measures at different layers of the cloud. [4] proposes a set of templates to cloud users, each specifying the query type that can be executed with their tread-off in time and cost. The client chooses among the proposed templates the one that best corresponds to her requirements. [5] proposes a set of matching algorithms to identify compatible cloud providers for a given requirements specification by matching SLA parameters, represented in RDF. RDF definitions are then converted into graph representations. An *Induced Propagation Graph* is calculated from both models to establish correspondences.

Recent works on SLA are devoted to extend SLA specifications to include security concerns. [6] propose an extension of the WSAgreement, initially developed by the GRAAP working group. Security constraints are expressed over the service description terms (SDTs) and the service level objectives (SLOs) of the SLA. This leads to an interoperable security expression that can be used by users for comparing security levels of cloud providers. Hale and Al. in [7] extend their first work proposing an ontology representation of security concerns and connect security services expressed in SLA to documents regulating security controls. [8] focuses on how to build a SEcSLA template starting from gathering then categorizing a set of security statements using a semantic tool. The designed template is then used both to express user security requirements and cloud service provider security provision. Finally, some works deal with SLA violation anticipation and SLA failure cascading on violation detection. [3] proposes an SLA dependency model using Web Service Modeling Ontology (WSMO) to build a knowledge database. [9] proposes to anticipate failure by analyzing the monitored feature. [9] proposes LAYSI, a layered solution that minimizes user interactions with the system and prevents violations of agreed SLAs.

Recent work on SLA modeling in the cloud does not seem to use SLA for data integration, in our opinion because measures concern data services and not data itself. Furthermore, SLA are in general exclusively used in a mono-cloud context to express the conditions in which a user can expect the cloud to provide resources. In contrast, data integration in a multi-cloud context implies expressing and enforcing multiple granularity constraints in different aspects of SLA (users, services, clouds and inter-cloud environments). Finally, an a-priori SLA enforcement mechanism - in opposition to the a-posteriori SLA monitoring to detect

violation- is mandatory to assess data integration feasibility. [10] proposed an e-auctions based approach for matching services with requirements in multi-cloud settings. Our work aims at going further in the use of SLA for integrating data and it provides strategies to guide the whole data integration process.

III. How should SLA be exploited?

To answer this question, we propose an SLA guided, security-aware continuous data provision and integration approach performed within a multi-cloud service oriented environment (see Figure 1). We present a step-by-step data integration process and the implied SLA categories.

In our context, data are provided by services deployed on different clouds. In the previous MOOC system example, several producers will supply equivalent content for a given period of time. They will be chosen with respect to the preferences expressed by a consumer.

Consumers specify queries as Datalog- or SQL-like programs including spatio-temporal attributes and preferences. For instance, the following query Q_1: "*List of English poetry content providers that can provide commented Emily Dickinson poems that are close to my city and are labeled as experts, where the total cost is less than 1 dollar, using only services that preserve their anonymity*".

The Abstract Inter-Cloud Layer (ICL, Figure 1) is responsible to deliver the results to the query. The ICL first identifies a set of abstract functionalities and associated services that fit the query. These descriptions are stored in a service description registry.

Assume that there are four content providers on English poetry that can be queried individually and two hubs that collect information from other sources like social network groups and hash tags. Hubs store content about given topics on English poetry, available from particular producers (e.g., participants of a given course). We represent these providers respectively by e_1 - e_4, h_1, h_2. We suppose that hubs have the same interface as individual service providers. We also suppose the existence of two free location services exporting the following interface: loc(IP) $\rightarrow \langle$ X, Y$\rangle$, meaning that given an IP address it returns a geographic position. All these services can potentially be combined for answering queries. Each combination will answer the query and fulfill in a specific degree the QoS requirements of the query.

Each deployed service exports an *agreed SLA* that specifies the economic cost per call, the maximum number of calls that can be done per day, the availability of the service, the average response time when a method is called, the reliability, the privacy of the produced data (whether they can be stored or not), the precision, freshness and provenance of the produced data, as defined below:

- agreedSLA$_i$: $\langle$cost, availability, freshness, provenance, data access control, certified reputation level, location, duration$\rangle$.

Some of these measures (cost/call, maxCall/day) are static and explicitly specified by the service provider. In contrast, the other measures should be computed by monitoring the conversations between the service and the applications that contact it. It is the case of user privacy-preserving service that will determine the way personal user data will be anonymized (Constraint1). Some measures will be instantiated depending on the context of the service invoked within a multi-cloud environment. It is the case for constraint 2 expressed above: user data access credentials should be calculated according to the policies adopted by the clouds hosting data services. An *agreed SLA* is given as an XML document using the specification WSLA (Web service level agreement[1]).

In our scenario, an example of computed measure is the cost of retrieving "expert" Emily Dickinson poems providers within a region, with their cost. The cost is determined by the cost of the calls. This request includes the price of calling a service (e.g., between $0,25 - 0,50, depending on the data service), plus the price of data transmission (amount of transmitted MB through the network), the type of subscription of the user for using the network and also the cost of applying encryption algorithms if required. The second computed measure is service reputation that is calculated according to the feedback obtained when application contacts the service.

As aforementioned, an *a-priori* SLA enforcement mechanism is mandatory to assess data integration feasibility. To this end, non-functional characteristics of the content provision services (i.e e_1-e_4) are calculated by specific cloud services that enforce SLA and which should export an API, composed of the following methods:

- estim-cost$_i$(contents, req_size, cost, prov_size, loc), agreedSLA$_i$.
- engage$_i$(contents, req_size, payment), agreedSLA$_i$.

For each service e_i, the first method returns the cost of the data (in our example, the cost of a given course (content)), as well as its size and location, and a required minimum size. The second method is used to engage to a data service (in our example a content data service) and will be used if the service is retained for a composition to answer the query.

The ICL obtains the set of potential services that satisfy both user preferences and *agreed SLAs* associated to services. For instance, the query Q_1 has two parts: the one that corresponds to the content required by the user that will be solved by combining data services and the *user preferences*, given by QoS preferences$_{user}$:

- QoS preferences$_{user}$: $\langle$cost, freshness, provenance, location, duration, privacy-preserving$\rangle$.

Let us suppose that for Q_1 the user is ready to pay a total cost of maximum $1; she requests that content providers

[1]http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf

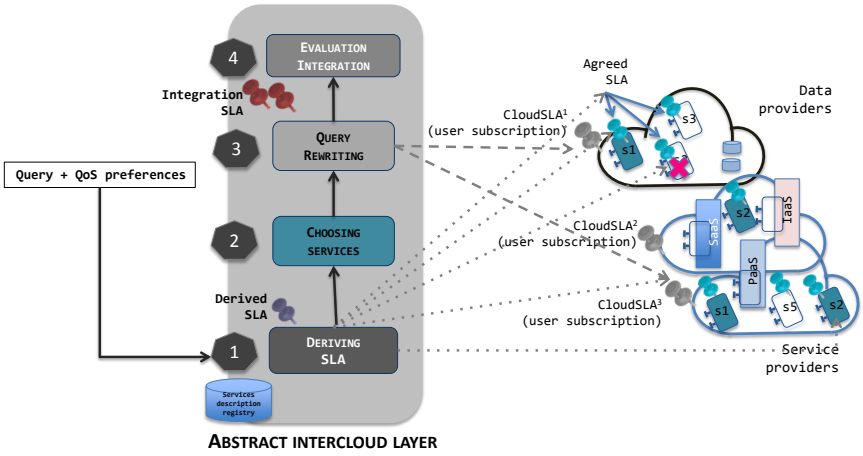

Figure 1. SLA guided data integration workflow.

be certified as experts (provenance) even if their data are not fresh; she requires to receive content during one week. Besides, she wishes to keep the content exchange private and she wants to order services by reputation. The maximum total cost will also determine the kind of privacy preserving algorithms that should be applied.

- QoS preferences$_{user}$: ⟨cost $\leq$ \$1, freshness = "any", provenance = "certified", location = "close", duration = 7 days, privacy-preserving="reputation-based"⟩.

Cloud providers also define their SLA contracts that specify the cost per request (cost/request), the volume of data that can be exchanged per month (I/0 volume/month), the cost of transferring data or applications within the same data center or across other data centers (datatransferCost/region), the storage space (storageSpace). For example, some cloud providers let the customer choose the geographical zone to install PaaS services and deploy applications (e.g. zone 1 is Europe). If the customer wishes to deploy services in zone 1 but store data in zone 2 the transfer cost will change.

- cloudSLA: ⟨cost/request, I/0 volume/month, datatransfer-Cost/region, storageSpace⟩.

Illustration inspired in the lowest contract provided by Azure[2]:

- cloudSLA: ⟨0,05 cents per call, 8 GB I/0 volume/month, free, 1 GB storage⟩.

The next step supervised by the ICL consists in elaborating a set of potential composition specifications via a query rewriting process. Given the user query, specified as a set of abstract operations and the available services, specified in the same way, we use a Local-as-View approach [11] to obtain a combination of services that match the query specification. The algorithm in [11] is being modified to deal with agreedSLA and QoS user preferences.

In this paper, we are interested in the process of rewriting queries considering QoS user preferences and SLAs. This process includes the following phases:

1) User preferences (including cloud SLA according to user subscriptions) and *agreed SLA* are used to produce a *derived SLA* to be associated to the query. *Derived SLA* influences the choice of services that will be used to build the result;
2) Computing service compositions that rewrite the initial query and that are used to build query results. The *agreed SLA* exported by the chosen services is added as conditions in each re-written query expression. Rewritten queries expressed as service composition can fully or partially comply with the *derived SLA*. An *integration SLA* is computed for guiding negotiation (in the case of partial compliance) and the implementation of the final integration;
3) Integration process guided by the *integration SLA*.

IV. Computing a derived SLA

The key and original aspect of our proposed data integration and provision process is defined as a vertical mapping of user QoS preferences and agreed SLA of potentially composable services. This leads to a *derived SLA* that guides the evaluation of a query and maps SLA measures and QoS preferences attributes.

The *derived SLA* is defined as a set of measures that correspond to the user preferences computed as a function of different static, computed and hybrid measures. These measures can come from the evaluation of simpler measures expressed in the cloud subscription of the user. The *derived SLA* will guide the way the query will be evaluated, and the way results will be computed and delivered.

In the example, some of the user preferences statement measures are used for defining a *derived SLA* that, as

[2]Azure is a trademark of Microsoft Corporation.

said before, will guide the evaluation of the query. These measures are defined as a function of the measures used by the agreed SLAs and by the cloud SLA contract. For example:

- total cost: $\sum_{i=1...n}$ cost(s$_i$) + data transfer + encryption cost $\leq$ \$1;
- availability: *(of services involved)* $\geq 90\%$; - freshness: non;
- provenance: all services involved must be $expert$; - duration: 7 days;
- I/0 volume/month: 8GB; - reputation level: $\geq$ threshold;
- storageSpace: 1GB.

Therefore, we propose a classification of SLA measures that represents the combination of fine grained measures used by *agreed SLAs* and coarse grained measures used in user preferences statements. It specifies also how to compute coarse grained measures with fine grained ones. For example, data precision will be computed as a function of availability, freshness and provenance exported by data services. The *derived SLA* can be seen as a set of inequations that have to be solved during the execution of a service composition. Since some of them can only be determined at execution time, the decision on which services will participate in the evaluation of the query is approximated. This latter step may conduct either to the rejection of integration in case of total incompatibility, or to a negotiation step between *agreed SLA*s and the *derived SLA*. In order to address this problem we introduce the notion of *integration SLA* (defined in Section VI).

V. Query Rewriting

Query rewriting consists in transforming an abstract query into lower-level queries that can be answered by available databases. The answers to the lower level queries are combined in order to obtain the result. The query rewriting problem can be generalized to the case of services: The query to be rewritten is seen as an *abstract* service composition, to be expressed in terms of *concrete* services. Query rewriting techniques have been adapted to the context of data providing services [12], [13] as well as for general service compositions [11]. The latter case (*i.e*, services that maintain and update information) is a generalization of the data provision case. 'The rewriting process is guided by the specification of both the query to be rewritten and the available services. The approach consists in generating translations of an abstract query into *several* compositions over concrete services. The solutions proposed are ranked and may be coded into concrete workflows that will be executed in one or several clouds by a special engine.

The approach proposed in [11] is being extended to the case of SLA-guided service-based data integration. Let us consider the MOOC scenario, where people can participate on a content trading pool. We suppose that this hypothetical MOOC has a large number of participants that sell their content to consumers. The MOOC has four information hubs described by their *agreed SLA*s, located in four different geographic locations, connected to other content providers. These later may be contacted directly, through the services of three different cloud providers that expose their *Cloud SLA*.

When an on-line consumer searches servers to buy/retrieve content, a composed service is computed. Depending on the location of the consumer and producers, different conditions and constraints may apply. These conditions (such as authentication or security requirements) should be also expressed in the QoS user preferences and *agreed SLA*. In this context, the user expresses a content order, her location and payment information. A composite service is generated to fulfill the order. The generated service composition should consider the nearest providers, in accordance to the *derived SLA*.

It is worth to notice that the algorithm in [11] shows an exponential behavior, since it produces all combinations of concrete services that satisfy the abstract specification. In order to solve this problem, we propose to use the *agreed SLA* and user preferences to guide the choice of concrete services, to allow that the most preferred ones are combined first. This will allow to produce the top *k* rewritings first (in accordance to the user preferences), thus avoiding the cost of producing all the solutions.

The *agreed SLA* is included and merged with other SLA conditions that may be obtained by the rewriting process. Only those compositions that comply with the *derived SLA* will be presented as possible solutions to the user. In the case of our query example the following composition can be used for answering it:

- Q(myIPaddress, "E.Dickinson", 1MB, "expert", \$1, myCreditCard) $\equiv$
  myLoc = loc(myIPaddress),
  estim_cost("E.Dickinson", 1MB, cost, size, theirLocation),
  query total cost + cost $\leq$ \$1, availability $\geq 90\%$, freshness = any,
  provenance = "expert", duration = 7 days, storageSpace $\leq$ 1GB,
  I/0 volume/month $\leq$ 8GB, reputation level $\geq \lambda$ (the threshold)
  engage("E.Dickinson", size, myCreditCard).

Our approach generates a number k of service compositions, combining as much as possible the services available such that the constraints of the *derived SLA* are verified. We are working on the modified version of the algorithm in [11] to take into account the different SLAs.

VI. Deriving an Integration SLA: first approach

Our data provision and integration approach relies on data services deployed on one or different cloud providers and it is delivered as a distributed DaaS on top of those clouds. The use of resources from a cloud is guided by an economic model (stated in a cloud subscription) that puts a threshold on the amount of resources used for query evaluation. It is thus important to optimize the use of these resources.

The optimization of this process can occur at two levels: first at the level of the *agreed SLA* exported by services. Indeed, queries requesting the same service compositions may have clauses that are conditions of use of the infrastructure (i.e., not storing the data produced by a service). Instead

of recomputing the *derived SLA* every time, we propose to store it and reuse it in future queries. Second, pre-computed queries and partial results can be also stored in cache or in a persistent support. Storing or not these results depends on the cloud subscription SLA of the user (data access, intermediate storage capacity, storage cost, etc.).

VII. Conclusions and Future work

This paper introduces the challenge of integrating data from distributed data services deployed on different cloud providers guided by service level agreements (SLA) and user preferences statements. The data integration problem is stated as a continuous data provision problem that has an associated economic cost and that uses automatic learning techniques for ensuring different qualities of delivered data (fresh, precise, partial).

Current big data settings impose to consider SLA and different data delivery models. We believe that given the volume and the complexity of query evaluation that includes steps that imply greedy computations, it is important to combine and revisit well-known solutions adapted to these contexts. We are developing strategies and algorithms overviewed in this paper and applying them to energy consumption applications and to elections and political campaign data integration to guide decision making on campaign strategies.

The diversity of mentioned application domains highlights the difficulty regarding SLA manipulation. Indeed, cloud, service, and user SLAs could be expressed differently in term of clauses, formalism and content. The multi-cloud environment emphasizes this heterogeneity.

The main issue when launching an integration process is to obtain the most accurate answer meeting the best user preferences. One of the possibilities when studying the integration is to accept the integration process in an incremental manner so that the query will continue to be evaluated till satisfying the set of QoS and non functional constraints expressed by the user. To our knowledge, the incremental production of a solution is beyond the scope of the current methods for rewriting service compositions and represents a challenge to the area. Furthermore, the integration SLA generation step is outstanding study. In fact the integration SLA should indicate the negotiation result with the aim to reuse it in further integration.

Acknowledgment

This work was partly funded by CNPq (Brazil, PDE-201118/2014-9, PQ-305619/2012-8, INCT-INES-573964/2008-4) and CAPES/CNRS (Brazil/France, SticAmSud 052/14).

References

[1] N. Bennani, C. G. Guegan, M. A. Musicante, and G. Vargas-Solar, "Sla-guided data integration on cloud environments," in *2014 IEEE 7th International Conference on Cloud Computing, Anchorage, AK, USA, June 27 - July 2, 2014*, 2014, pp. 934–935. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2014.130

[2] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments," in *HPCS 2010*, 2010, pp. 48–54.

[3] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "A dependency-aware ontology-based approach for deploying service level agreement monitoring services in cloud," *Softw. Pract. Exper.*, vol. 42, no. 4, pp. 501–518, Apr. 2012.

[4] J. Ortiz, V. T. de Almeida, and M. Balazinska, "A vision for personalized service level agreements in the cloud," in *Proc. 2nd Workshop on Data Analytics in the Cloud*. ACM, 2013, pp. 21–25.

[5] T. Chauhan, S. Chaudhary, V. Kumar, and M. Bhise, "Service level agreement parameter matching in cloud computing," in *Information and Communication Technologies (WICT), 2011 World Congress on*, 2011, pp. 564–570.

[6] M. Hale and R. Gamble, "Secagreement: Advancing security risk calculations in cloud services," in *IEEE SERVICES*, 2012, pp. 133–140.

[7] ——, "Building a compliance vocabulary to embed security controls in cloud slas," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*, June 2013, pp. 118–125.

[8] J. Luna Garcia, R. Langenberg, and N. Suri, "Benchmarking cloud security level agreements using quantitative policy trees," in *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, ser. CCSW '12. New York, NY, USA: ACM, 2012, pp. 103–112. [Online]. Available: http://doi.acm.org/10.1145/2381913.2381932

[9] I. Brandic, V. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertesz, and G. Kecskemeti, "Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures," in *IEEE COMPSACW*, 2010, pp. 365–370.

[10] M. Anisetti, C. A. Ardagna, P. A. Bonatti, E. Damiani, M. Faella, C. Galdi, and L. Sauro, "e-auctions for multi-cloud service provisioning," in *Services Computing (SCC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 35–42.

[11] U. S. da Costa, M. Halfeld-Ferrari, M. A. Musicante, and S. Robert, "Automatic refinement of service compositions," in *ICWE*, ser. Lecture Notes in Computer Science, F. Daniel, P. Dolog, and Q. Li, Eds., vol. 7977. Springer, 2013, pp. 400–407.

[12] M. Barhamgi, D. Benslimane, and B. Medjahed, "A query rewriting approach for web service composition," *IEEE T. Services Computing*, vol. 3, no. 3, pp. 206–222, 2010.

[13] W. Zhao, C. Liu, and J. Chen, "Automatic composition of information-providing web services based on query rewriting," *Science China Information Sciences*, pp. 1–17, 2011.