Abstract

The first part of the paper describes the design of a service module that assists customers with the leasing of cloud computing services by taking into account their requirements on service provisioning. The service module introduces basic service level management functionality within a larger framework. The framework is illustrated as a web platform and uses a trusted third party (TTP) as a secure middle layer on service transactions. The second part of the paper focuses on the TTP role. Preliminary security requirements are identified and analyzed for the implementation and successful operation of such a TTP entity.

Reference


Available at:
http://archive-ouverte.unige.ch/unige:23362

Disclaimer: layout of this document may differ from the published version.
Preliminary requirements on trusted third parties for service transactions in cloud eco-systems

Katerina Stamou*, Jocelyn Aubert†, Benjamin Gateau† and Jean-Henry Morin*

*Institute of Services Science, University of Geneva - HEC, Switzerland
Email: aikaterini.stamou, Jean-Henry.Morin@unige.ch
†Service Science & Innovation, Public Research Centre Henri Tudor, Luxembourg
Email: Jocelyn.Aubert, Benjamin.Gateau@tudor.lu

Abstract—The first part of the paper describes the design of a service module that assists customers with the leasing of cloud computing services by taking into account their requirements on service provisioning. The service module introduces basic service level management functionality within a larger framework. The framework is illustrated as a web platform and uses a trusted third party (TTP) as a secure middle layer on service transactions. The second part of the paper focuses on the TTP role. Preliminary security requirements are identified and analyzed for the implementation and successful operation of such a TTP entity.

Keywords—service level management, SLAs, cloud computing, trusted third party, service broker

I. INTRODUCTION

The scope of this work is twofold. The first part of the paper initializes our discussion with a high-level description of the CLOVIS framework [8] and a brief summary of current service level management approaches. A framework overview introduces its internal components and in particular the SLAaaS module.

To date, in cloud computing, service level agreements (SLAs) are used to represent service availability rates through provider guarantees. Currently contract terms are written in human language, which prevents their automatic generation, update and monitoring. One goal is to use SLAs as they are used in grid computing environments, but represent them as e-contracts through a web interface.

The SLAaaS module operates within the CLOVIS framework and attempts to realize the aforementioned goal. The module receives as input pre-initiated SLAs in the form of service offers and enables customers to select an adequate SLA based on how they would like to have their leased service provisioned. The design analysis of the SLAaaS module includes a component-based description, the core service module protocol, design trade offs and implementation details.

The second part of the paper focuses on the role of TTPs for service transactions in cloud environments. The CLOVIS framework utilizes a TTP entity as a secure middle layer to manage the interaction between service customers and providers.

We elaborate on how the TTP role can be conceived within the cloud computing paradigm and briefly review background literature work on TTPs and cloud brokers. We analyze identified security requirements and propose how such requirements can be realized when utilizing the TTP notion for services that operate over non-trusted networks like the Internet.

We complete our analysis with a simple approach that can enhance the TTP conformance to its role and responsibilities. This approach illustrates the SLA model that the TTP needs to provide, when its role is to manage client data transactions over an open cloud service. We conclude the paper with final remarks and current work.

II. CLOVIS FRAMEWORK DESIGN

A. Framework overview

In [8] we elaborated on a framework where risk analysis’ results map to SLOs and define service consumer requirements. SLAs express Services Level Objectives (SLOs) that a service provider is committed to provide. SLOs accurately indicate the capacity capability of a service provider.

As depicted in Figure 1, our high-level model assumes the existence of a risk management (RM) tool, which can provide a customer with an adequate risk assessment according to a customer’s service criteria and requirements (RMaaS). Such service requirements can then be utilized as input to correctly match with available service offers, i.e. pre-instantiated SLAs that are submitted by service providers. SLOs and risk controls can be manipulated for this purpose.

Figure 1. Risk Management - SLA coupling
In the use-case scenario described in this paper, the framework components are identified within a client-server architecture and data exchange takes place over the HTTP protocol. The server exposes information to the client part via service module interfaces.

The server is primarily comprised of the SLAaaS and RMaaS modules, which both implement specific business logic. These two modules are inter-connected in that RMaaS receives data input from SLAaaS and vice versa. An overall framework view is illustrated in Figure 2.

![Figure 2. CLOVIS framework overall view](image)

Figure 2. CLOVIS framework overall view

The trusted third party (TTP) entity is also hosted on the server part. The TTP acts as a secure middle layer in the information exchange between transacting parties and coordinates the process of transactions in terms of data integrity and privacy. The TTP updates and maintains a user management repository to hold information about users and their activities.

A service offer repository, based on a relational DBMS, receives constant input and feeds back output to the SLAaaS framework module. The RMaaS module sends and receives data from SLAaaS.

**B. Existing service level management approaches, motivation**

One idea behind the CLOVIS framework is that the framework should be a compatible tool with any backend infrastructure that does service level management and offers monitoring facilities. With respect to service level management we have focused on the WS-Agreement specification [1] and elaborated on how its utility in terms of both protocol and language structure can fit into our research objective.

The Grid Resource Allocation Agreement Protocol (GRAAP) working group [7] has proposed the WS-Agreement specification as a structured method on service level management for distributed systems. The specification has been efficiently utilized for SLAs execution on a machine interpretable, automated way. Such execution is needed for the scheduling and running of advanced computing tasks that run on distributed platforms.

The WS-Agreement specification follows an object oriented design, is based on the XML schema and utilizes the web services architecture stack. A Java based implementation of WS-Agreement and WS-Agreement Negotiation [11] is WSAG4J. WS-Agreement introduces a consistent and efficient approach to SLA management. Its usability has been proved by its widespread application on distributed computing infrastructures and scientific computing tasks.

On the other hand, one of CLOVIS design requirements is the SLA exposure in a both human and machine friendly manner such that SLAs can become utilizable on a business level and regardless of the application domain. Our SLA format structure inherits many elements and ideas from the WS-Agreement specification, but does not implement it. CLOVIS introduces a ReSTful API, where a service offer structure and internal components are merely described in JSON terms.

**III. CLOVIS SLAAAS FRAMEWORK MODULE**

This section describes the SLAaaS module design. This module illustrates the core CLOVIS framework functionality and represents the integration link to all other framework modules. Figure 3 depicts a component-based aspect of the SLAaaS module and its interaction with other framework components. In the scenario that we discuss in this paper, the TTP acts as transaction coordinator and proxy to secure the communication exchange.

![Figure 3. SLAaaS component based overview](image)

Figure 3. SLAaaS component based overview

Service offer formulation is the component that allows service providers to formulate their offers. Service providers typically create SLA offers in the form of pre-activated agreements to indicate their service capacity and advertise the type of QoS they can provide. The service offer structure and many internal offer terms are derived from the WS-Agreement specification. The service offer formulation is based on the JSON format.

We try to expose the editing, creation and publishing of service offers through a User interface. All offers that are submitted in the SLAaaS framework are public. A service provider can submit one or more offers. Service customers

1WS-Agreement for Java, [http://packcs-e0.scai.fraunhofer.de/wsag4j/](http://packcs-e0.scai.fraunhofer.de/wsag4j/)
retrieve offers through this interface with help of the *filter-match mechanism* and submit their agreement preferences.

Submitted offers and initiated agreements are stored in the SLA template/SLA repository. This component keeps information on evaluated service offers. Such information describes service offer internal terms and exposes service offer guarantee values that in turn are provided as input to the filter-match mechanism. The repository structure is driven by object oriented design and a service offer is treated as the primary object. MySQL is used as the backend relational DBMS.

The *filter-match mechanism* represents the core business logic that the SLAaaS service module implements. The mechanism filters customer requirements and matches them to submitted offers, if matching offers exist. Service offers are decomposed into fine-grained components and mapped into the data management layer to enable term-based queries. The matching filters are realized based on these queries.

*Monitoring* on active SLA status is identified as an external module component. Its integration is planned through third services that implement such functionality. The WS-Agreement specification proposes an interface that enables the monitoring of active agreements. In the service module, the TTP entity is responsible for exposing the agreement status to the associated client part through the user interface.

Similarly, the *value added services* module represents a ReSTful API that allows integration of new services and adapts existing ones into the CLOVIS framework. Risk analysis tools, auditing and accounting modules are such service candidates.

The *filter-match mechanism* abstracts the direct need to negotiate service offers. With the filter-match usability a customer can initially select and decide for service offers that meet their expectations on service provisioning. Negotiation between service providers and customers, if applicable, can take place after a limited set of service offers is selected. The service module process on negotiation handling is not analyzed in this work.

### A. SLAaaS module protocol

The WS-Agreement specification defines the roles of involved parties in the initiation of the SLA management cycle. According to the specification, a service customer is typically the agreement initiator and a provider the agreement responder. The agreement initiator selects from submitted service offer templates and sends a counter offer to the agreement responder. The agreement responder accepts or rejects the initiator’s request.

The SLAaaS module adopts similar logic. It is assumed that service providers are bound to what they submit as service offers, since the application layer is on the upper business level. If a customer wishes to negotiate with the provider on the service provisioning levels, the provider needs to declare negotiation applicability beforehand.

Figure 4 illustrates the SLAaaS communication protocol as a sequence diagram between the TTP and involved parties. This illustration covers the scope of the use case scenario that we discuss in this paper. The TTP has the role of the intermediary in the two parties communication.

![SLAaaS communication protocol](image)

A service provider submits one or more service offers to the SLAaaS module, as they would submit a service description to any marketplace or UDDI like registry. A service offer can be submitted either through an automated process that utilizes the client module API or as a stepwise process through a user interface. The service module evaluates the offer in terms of structure and anticipated terms. There are no restrictions as to the number of service offers a service provider may submit.

A service customer selects among existing service offers with the help of the filter-match mechanism. Customers submit their service provisioning requirements and the mechanism matches such criteria to existing offers. Customers cannot impose new requirements. Instead, upon selection they may request to negotiate with one or more providers based on providers’ negotiation applicability.

Otherwise a customer is directed to the selected provider’s marketplace to get more information about the service characteristics and potentially purchase the service. Alternatively, a customer may request to initiate an agreement with the selected service provider instantly. In this case a customer may request from the SLAaaS module to manage the initiated agreement if they don’t plan to use an in-house management mechanism.

### B. SLA formulation, filters & matches

The scope of the filter-match mechanism is to expose service properties and guarantees that match with customer service provisioning requirements.

To achieve this, we have decomposed the SLA structure as defined by the WS-Agreement specification into granular,
utilizable components. We utilize such decomposition to retrieve a service offer model that can be exposed through a user interface. Alternatively, a service offer can be submitted by a client automated process. A client API, based on the ReST architecture and the JavaScript Object notation, serves this purpose.

Moreover, WS-Agreement introduces the concept of agreement factories that hold offer templates. Our design adopts this concept to classify service offers according to their service type and application domain specific criteria. The SLAaaS module evaluates a submitted service offer according to its corresponding template, in terms of inner structure and expected terms.

Submitted offers are stored in the service offer data repository. Following the structure of the WS-Agreement specification, the primary components that we are interested in are the service offer context and service guarantees. According to WS-Agreement, the SLA context section includes information that does not reveal specific obligations in terms of service properties. The context section can be extended to include information like a service provider’s infrastructure location, the location where customer’s data is going to be kept or a provider’s legislative authority. Such QoS information may be of vital importance for a customer decision on how they would like to have their service provisioned.

In addition, according to WS-Agreement, guarantee terms may either apply to the overall service where the SLA applies, or to specific service properties. We distinguish the assortment of guarantee terms between these two types. Guarantee terms that apply to specific service attributes are stored along with their referenced properties and corresponding metrics.

The WS-Agreement specification includes an extra section of creation constraints that apply to agreement templates only. The agreement constraints section describes restrictions on service capacity levels that a provider can correspond to. The SLAaaS module includes such creation constraints section for service offers, where SLA negotiation is applicable.

Customers specify the service type that they are interested to lease. According to this input, they can then adjust their provisioning requirements based on existing offers availability. This process takes place through the user interface and resembles the “airplane-ticket” discovery process that many aggregator services currently offer. The mechanism filters existing offers based on submitted customer requests, which are translated into queries, and returns available matching results.

C. Design trade-offs, usability aspects

CLOVIS application domain is focused on cloud computing services at the infrastructure layer. One research objective has been to couple SLAs with RM in order to assist the decision making process of customers when investing into new service products. Service providers typically follow RM and decision making analytics during and before their service offer formulation. Receiving input from customers on how they would like to have a service provisioned introduces additional criteria on service offer formulation and SLA management.

We have investigated three primary use-cases where the SLAaaS module can apply and prove a highly usable tool. First and foremost, the functionality that the SLAaaS module proposes can integrate with service marketplaces or registries as an additional decision making tool during service selection process. Providers can improve their service offer formulation by utilizing statistical data that are derived from the tool usage in terms of how customers seek a service to be provisioned.

Moreover, the SLAaaS module can function within inter-organizational environments to assist service designers and developers with the formation of new service products and provisioning approaches. The IT virtual economy requires new and effective methods of service provisioning.

We also assume that the process of service selection, based on service provisioning criteria and not only service content attributes, can stand as a standalone web service within a broader service level management framework. This is the use case that we discuss in this paper.

Additional design requirements that this latter use case introduces, deal with user data management. So far, we excluded from our description how user data are handled in the framework to assure privacy and user confidence. We needed to provide first a description of the designed service module in terms of scope and functionality. In the rest of the paper we analyze identified security requirements. Most such requirements deal with management of user sensitive information. We utilize the TTP notion as the managing entity that handles such data and enables secure user transactions.

Last but not least, the described scenario introduces several performance challenges. Such requirements will be analyzed on future work based on experiments with the developed service prototype.

IV. Trusted Third Parties in Cloud Computing

The origins of the TTP mechanism are found in cryptography, where a TTP represents an entity that allows secure (trustful) interaction between two distinct parties. It is assumed that both parties consider this third entity a trusted one. A widely used TTP implementation has been that of certificate authorities for web transactions.

Outside of strict cryptographic bounds a TTP can be perceived as a public notary service. According to ISO [4] a TTP has to implement security policies that prevent unauthorized usage of resources by any involved party. In [6] NIST defines cloud actors that typically interact within a cloud eco-system. According to the NIST architecture,
service customers can utilize broker services instead of direct interaction with providers to manage cloud products.

By NIST definition a cloud broker represents an entity that handles how a cloud service performs, is utilized and dispatched to one or more users. In addition a cloud broker can also assist the role of a negotiator between service providers and customers. Cloud brokering services are mapped into three general areas: intermediation, aggregation and arbitrage.

Consequently, a cloud broker entity can also be realized via a TTP to enhance security and trust in cloud transactions. By definition a TTP acts in a neutral manner as a mutually trusted entity. An intermediary can be neutral and operate based on predefined rules with the goal to balance trust and confidence among transacting parties. A public notary in the cloud can be characterized as a neutral intermediary that enforces a set of security policies.

Thus, our TTP notion extends the traditional TTP definition. The TTP entity operates within a service level management framework to orchestrate the service offer submission and to secure communication and information handling between transacting parties.

V. TTP PRELIMINARY SECURITY REQUIREMENTS

A TTP model that orchestrates the interaction between contracting parties in an open service level management framework is a flourishing approach. This section analyzes initial security requirements that need to be realized in order to justify the TTP role.

According to ETSI [2] a TTP entity has to provide all mechanisms necessary to ensure confidentiality and integrity of information handling between two parties. From a user perspective a motivation to utilize a TTP is to enhance trust and confidence in their business transactions. A TTP may be perceived as an assurance layer on facilitating secure communication between parties.

By default, a TTP entity implements necessary security mechanisms that enable robust user authentication and authorization access. The TTP assigns user privileges according to user operations. A user may have an operational role both as service provider and customer. All users must conform to the same TTP rules.

ETSI recommends that a TTP scheme implementation should follow an assessment on identified risks and security requirements regarding specific services the TTP is about to provide. Such risks indicate cases where the TTP may be compromised, while security requirements assist in specifying security policies that the TTP scheme should implement.

A. Confidentiality

The TTP must ensure at all costs confidentiality of sensitive information that is stored or exchanged within framework bounds. In a service level management framework sensitive information may involve stored service offers, possibly initiated agreements and data from monitoring of such agreements’ states. Sensitive information includes any client data, whose disclosure would impact either or both concerned parties, including the TTP conformance.

To this end, the TTP has to protect data that is persistently or temporarily stored under its control bounds. In addition, the TTP has to implement safeguards to preserve non-tampered information that is exchanged over the Internet and ensure data partitioning to avoid unauthorized data access. Usage of public key cryptography and access control mechanisms constitute seasoned approaches to meet these requirements.

B. Non-repudiation, information traceability and auditing

The TTP has to guarantee non-repudiation of exchanged information. In a service level management framework, information is reflected through resources that may concern operational and private data. According to ISO 7498-2 [4] repudiation takes place when one of the two parties involved in a transaction denies its participation in all or part of the communication process.

SLAs include guarantee terms that describe boundaries and thresholds on service properties and availability of service capacity. Adherence to such terms is verified by monitoring the resource state that reflects such service levels during runtime. Typically, a service provider is responsible for monitoring. A customer may request audit operations from an external entity.

ISO/IEC 10181-7 [5] identifies non-repudiation mechanisms and techniques that may be adopted by a TTP scheme. The TTP generalization into a public notary has to include following mechanisms:

- Security tokens with respect to authentication and authorization of registered users. According to the Open Group [10] single sign-on is a mechanism that allows users to access resources (remote systems and services) without need of multiple logins. Such mechanism is useful in case a user needs instant contract initiation and direct access to agreed services.

- Evidence retrieval and verification is a cornerstone in providing non-repudiation services. According to ETSI the term evidence corresponds to the technical instrument used to provide proof. The WS-Agreement specification provides a monitoring interface that encapsulates every possible agreement state.

An internal timestamp mechanism can be used to maintain data logs with respect to transit of private information. Timestamps can be derived with a signature or a unique identifier affix to a data item at predefined time intervals. Moreover a message digest and a hash algorithm can be used to prove the creation of information (e.g. customer agreement request) at a given time point. A service level management framework can store a log of such evidence to be used in case of disputes or other conflict.
C. Integrity

The TTP has to be tested and verified against its operational integrity. It has to assure that any system functions and service operations that the framework integrates, will not harm or impair user sensitive data and related information.

The TTP has to explicitly state security policies that it implements and how such policies may affect a party’s critical information. Intended use of such policies needs to be clarified as to how they assure secure functionality of relevant system operations.

The TTP has to specify how it guarantees non-modification and secure information transit within the service level management framework. It has to define concrete liability bounds regarding its intermediary role and prompt users to take additional precautions, if they need to.

Last but not least, both transacting parties need to be aware of all available options regarding monitoring and auditing. Every interested party needs to know the policies that allow an externally operated monitoring mechanism to integrate with the service level management framework.

D. Availability

Availability of service and user information is a fundamental security requirement for any system and scheme implementation. A registered user should be able to access all services exposed by the framework at any time interval that is defined in the TTP guarantee terms, and make use of all eligible operations.

Additionally, the framework must allow registered users to retrieve relevant information in a readable format regardless of the remote system, which they operate. Thus the platform that hosts the service level management framework needs to be compatible with any supported operating system and standardized data format that is explicitly defined in the TTP policy and guarantee declaration.

E. Building trust and confidence

The realization of a TTP scheme is merely based on the fundamental requirement that the transacting parties trust the TTP to perform operations. The TTP is considered a widely accepted, reliable, independent and highly secure entity that generates trust. Thus, it eliminates the need to maintain individual bilateral agreements regarding the authenticity and legitimacy of contracting entities [2]. This is an important advantage on the choice of the TTP scheme as a model for contracting services in the service level management domain.

Every business domain imposes their own rules with respect to trust levels and security mechanisms that their operations require. Accomplishing a trusted system may be considered a contrasted even subjective matter. Not all TTPs are able to create confidence, particularly while initializing their activities.

On the other hand, a clear definition of service and security assurances provides an initial trust layer. A user is aware of what the system guarantees and is bound to. System confidentiality can be described as information that is sensitive and private for one or more system entities. There are multiple approaches on how a TTP model can secure user confidential information.

The TTP scheme can implement its own certificate authority to produce temporary client certificates in conjunction to the SSO mechanism. Certificate authority and SSO mechanism can be integrated under a PKI [9] infrastructure. Produced certificates can follow the X509 certificate schema and provide an extra assurance layer as to the trustworthiness and identity claims of contracting parties.

Along with digital certificates generated or derived from established certification authorities, a recognized, commonly trusted notary entity has to affirm that the TTP is trustworthy. Using the notion of trust hierarchies and trust transfer, any entity that trusts the notary entity will also place its confidence to the TTP.

Given that this approach is again based on an entity that is considered trusted, we acknowledge its limitations. In this sense, our viewpoint is to enable hierarchically higher trusted entities to periodically audit the TTP. Such entities could be for example inter/national administrations whose confidence and independence are well established. The same trust transfer model can equally apply in an inter-organizational environment.

In addition, the TTP reliability can be validated by reputation mechanisms. Such mechanisms can operate externally. The framework can maintain an internal reputation mechanism and combine it with a dispositional trust model to enhance mutual trust and confidence between transacting parties.

VI. TTP CHALLENGES, CONFORMANCE ASSURANCE

A. TTP approach challenges

TTP models have received a certain amount of criticism - not unjustified - with respect to their usage and security requirements adherence.

As aforementioned, a TTP has to be accredited and verified by a commonly accepted authority that in turn may adhere to inter/national legislation and standards. Still, it is doubted that the TTP will conform to applied laws; or that it will orderly enforce its security policies. It is difficult to control the internal operations of a TTP and to guarantee its correct functioning.

Consequently, such obscurities enhance the levels of anti-trust towards the TTP scheme and impose stronger control mechanisms to verify its operations. Apart from strong logging and auditing mechanisms, there are not many options to control a TTP entity.

Moreover, it is difficult to pre-assume the willingness of service providers to open external ports in their systems and
allow resource access to third monitoring mechanisms. It is also doubtful that providers will submit accurate logs with respect to monitored values of their resources. Such tactics hinder the resolution of disputes between contracting parties.

B. TTP conformance

Typically, a TTP is an organization licensed or accredited by regulatory authorities and international standards to provide security services on a commercial basis and to a wide variety of users [2].

To validate our design approach using the TTP as coordinating entity in the CLOVIS management framework, we assume that in addition to a set of policy terms, the TTP has to provide an SLA that specifies TTP guarantees in terms of service operations that it supports and is liable to. Such SLA needs to apply for all prospective service users uniformly and irrespective of their role as service provider or customer.

This SLA needs to specify details on service availability along with integration options of externally operated services into the management framework. The SLA may also notify prospective users regarding the inter/national legislation and treaties that the hosting platform is subject to. Likewise many wide service providers today include in their agreements and policies their court district or state legislation.

Such an SLA is important regarding TTP conformance to agreed and guaranteed service objectives. It provides a fine way to control TTP adherence to its role and responsibilities. A user, who wants to utilize services offered by the management framework, needs to accept the conditions and terms that are specified in the SLA.

The SLA needs to provide uniform agreement content to any prospective user. In addition, such an SLA establishment needs to comply with open standards of service provisioning and allow for unanimous requests and service provisioning updates.

C. TTP SLA model

We provide a simple model of how the TTP supplied SLA may look like. There are two central characteristics in this SLA: neutrality and openness. Figure 5 illustrates the proposed model.

The TTP entity represents the service provider in this SLA. The agreement is between the TTP and the parties that the TTP serves. By default, the TTP represents the intermediary in the interaction between two parties ($TE_a$ and $TE_b$). In framework terms, the number of clients can be from 1 to $n$ for each contracting entity. In the SLA, the total number of clients from both sides is represented as a one-to-one relationship with the TTP entity.

Similarly a transacting party has a one-to-one relationship with another transacting party. In the same time an entity may have multiple transactions with different transacting entities. Thus, a transacting party can have 1 to $n$ business transactions within the management framework.

An initial assumption of using a TTP is that in general a TTP represents a neutral entity. The TTP does not operate in favour of any of the transacting parties. Accordingly all service description parameters and service level guarantees that are provided in the TTP SLA apply unanimously to any party.

Guarantee terms that the TTP accommodates are mainly provisioning terms regarding the service platform and bounds on individual party processes that may take place within the framework. The TTP SLA contains clauses that assist conformance to the enclosed services communication protocol.

The TTP SLA defines no rewards but merely penalties. The TTP entity is subject to most penalties, if there is evidence of non-compliance and illegal activities. Transacting parties are also subject to penalties if there is evidence of non-compliance to the commonly agreed protocol and set of policies. Penalties may be imposed in the form of reputation tracks. Often, reputation in business transactions is much stronger than any financial penalty.

Another crucial characteristic of the TTP SLA is openness. As mentioned by [3], SLAs are crucial in eBusiness models since they provide the means to compensate a customer’s dependency on a service provider. The TTP SLA can act both as control and formulation mechanism on how the transacting parties can interact. In other words, transacting parties can unanimously request modifications and updates on the TTP defined service conditions.

The TTP entity has to guarantee in the SLA open access to all its logs, audits and internal data in case a dispute occurs that may or may not involve the TTP. Thus, any transacting party may require access to query evidence within the TTP control boundaries in case a conflict occurs.

The TTP SLA has to determine within a properties section a concrete description of what, how and where security
policies are implemented and apply in the framework. It has also to provide an exact specification of how the proposed TTP SLA can be updated according to unanimous user votes.

VII. CONCLUSION & ON-GOING WORK

The first part of this work introduced the overall design of the CLOVIS framework. In particular, we described in detail the SLAaaS module inter-components, design and intended functionality. The SLAaaS module is positioned within a larger management framework - in our case CLOVIS - and introduces service level management operations that take place before an SLA instantiation.

The module receives service offers (pre-initiated SLAs) as input and helps customers to match their service provisioning requirements. On successful matching a customer may select to directly lease a service or, if applicable, to negotiate with the provider on service level terms. To do so, service offers are decomposed into granular components.

The SLAaaS module is currently under development; the implementation prototypes the web application that we described in this work. Development focus is currently turned on defining a generic but suitable method to express service offers in both a human friendly and machine readable manner. As aforementioned, we inherit many components from the WS-Agreement specification. Still, the JSON format has been selected over the XML schema notation.

In addition, we try to merge the definition of guarantee terms with their relevant service description properties in a manner that allows their efficient manipulation from the database following object oriented programming techniques. The web based prototype version relies on the ReST architecture. The PHP scripting language has been selected as the primary programming tool.

The second part of the paper discussed the role of TTPs in cloud computing environments. We primarily focused on a TTP entity that acts as a secure middle layer within the designed framework. The TTP needs to enable secure handling of user data and ensure confidence on user transactions, regardless of a client role as service customer or provider. The TTP has broker characteristics since it manages the transaction process between contracting parties. Still, it retains its neutral behavior.

We tried to cover multiple security aspects that generate requirements around a TTP’s role in cloud environments. Finally, we proposed an SLA model that any TTP can provide to its clients in order to increase its conformance and client trust on TTP operations. Many of the identified security requirements and the proposed TTP SLA model are included in the web prototype.

ACKNOWLEDGMENT

This work is supported by the CLOVIS project jointly funded by the Swiss SNF and Luxembourg FNR Lead Agency agreement; under Swiss National Science Foundation grant number 200021E-136316/1 and Luxembourg National Research Fund (FNR) grant number INTER/SNSF/10/02.

REFERENCES


