Federation Establishment Between CLEVER Clouds Through a SAML SSO Authentication Profile

Antonio Celesti, Francesco Tusa, Massimo Villari and Antonio Puliafito
Dept. of Mathematics, Faculty of Engineering, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {acelesti,ftusa,mvillari,apuliafito}@unime.it

***Abstract*—Cross-Cloud federation implies the establishment of a trust context between cloud platforms acting on different administrative domains and located in different places. The main advantage of federation is that clouds can set interdomain communications so that they can benefit of new business opportunities such as the enlargement of their virtual resources capability. The process of federation set up can be schematized in three subsequent phases: Discovery, Match-Making, and Authentication. In this work, considering several clouds based on both the CLEVER architecture and a Cross-Cloud Federation Manager module, responsible for the accomplishment of the three phases, we focus on the authentication phase required for a secure interaction between different CLEVER domains. More specifically, we designed a SAML SSO profile for a generic three-tier cloud architecture, showing the way in which it can be applied in different CLEVER-based clouds for the establishment of trusted interdomain communications in order to "lend" and "borrow" virtualized resources.**

***Keywords*-Cloud Computing; Federation; Authentication; CLEVER; XMPP; SAML.**

I. Introduction

Cloud computing brings a new level of efficiency in delivering services, i.e., Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), representing a tempting business opportunity for ICT operators of increasing their revenues.

Currently, the cloud computing scenario includes hundreds of independent, heterogeneous, private/hybrid clouds but, many business operators have predicted that the process toward an interoperable federated cloud scenario will begin shortly. According to Gartner [1], the evolution of the cloud computing market is hypothesized in three subsequent stages: stage 1 "Monolithic" (now), cloud services are based on proprietary architectures - islands of cloud services delivered by mega-providers (this is what Amazon, Google, Salesforce and Microsoft look like today); stage 2 "Vertical Supply Chain", over time, some cloud providers will leverage cloud services from other providers. The clouds will be proprietary islands yet, but the ecosystem building will start; stage 3 "Horizontal Federation", smaller, medium, and large providers will federate horizontally themselves to gain: economies of scale, an efficient use of their assets, and an enlargement of their capabilities. For simplicity, in the rest of the paper, with terms such as "cross-cloud federation", "federation in cloud computing", or "cloud federation" we will refer to the above mentioned "Horizontal Federation".

In our previous work [2], we described how to set up an interoperable heterogeneous cloud environment in a Horizontal Federation, where clouds can cooperate together accomplishing trust contexts and providing new business opportunities. In that work, we proposed a three-phase cross-cloud federation model where the federation establishment between clouds passes through three main phases: *discovery*, the cloud looks for other available clouds; *match-making*, the cloud selects between the discovered clouds the ones, which fit as much as possible its requirements; *authentication*, the cloud establishes a trust context with the selected clouds.

The authentication phase poses many serious problems in the cross-cloud federation establishment due to the need for each cloud of managing a huge number of credentials depending on the security mechanisms employed in each infrastructure. In fact, a cloud should be able to authenticate itself with other heterogeneous clouds regardless their security mechanisms, performing the log-in once, gaining the access to all the required resources. We identified such a problem as *Cloud Single-Sign On (SSO) Authentication* and we addressed it designing a new Security Assertion Markup Language (SAML) [3] profile, defining the steps needed for a secure cloud SSO authentication.

The current open source cloud implementations lack of modularity in their architecture, hence, it could be very difficult to modify them or integrating new features. For these reasons, at the Multimedia and Distributed Systems Laboratory (MDSLab) of the University of Messina, we have been developing a new Virtual Infrastructure Manager named CLoud-Enabled Virtual EnviRonment (CLEVER) [4].

In this work, starting from the analyzed issues regarding the Horizontal Federation, and considering our proposed solution for enabling the SSO authentication using SAML, we apply our idea to a concrete scenario including CLEVER-based clouds. CLEVER well meets the requirement of a cloud scenario whose actors intend to establish a Horizontal Federation: although it specifically aims at the design of a management layer for the administration of cloud infrastructures, differently from other existing open source middleware, it also provides simple and easily accessible interfaces

for enabling the interaction of different "interconnected" clouds.

The paper is organized as follows. Section II describes the state of the art of cloud federation. In Section III, we provide a detailed analysis of cloud federation requirements, introducing the concept of *home cloud* and *foreign cloud*. In section IV, we introduce our three-phase federation model, also discussing the Cross-Cloud Federation Manager (CCFM) module. In Section V, we focus on the authentication phase and in particular on the cloud SSO authentication problem, running through the SAML CCAA-SSO profile. Section VI provides the details about the CLEVER architecture, pointing out its main features. Section VII provides a detailed description of the SAML CCAA-SSO profile applied to a federation scenario including CLEVER-based clouds. Conclusions and lights to the future are summarized in Section VIII.

II. Related Work and Background

Cloud Computing is emerging as a promising paradigm able to provide a flexible, dynamic, resilient and cost effective infrastructure for both academic and business environments.

The paradigm is rather different if compared with the previous one, that is Grid computing, as it was remarked by Ian Foster, the father of Grid, in his work [5]. In particular consolidated research topics are appearing hard to face on cloud computing especially on the area of security.

As it is recently reported in [6], the authors have underlined that security and privacy in Cloud represent of the main challenges. They remarked in cloud environments it is necessary to deal with: authentication and identity management, trust management, policy integration, access control and accounting, secure service management, privacy and data protection, semantic heterogeneity. They recognized that cloud computing are becoming multi domain environments in which each domain can use different security, privacy, and trust requirements and potentially employ various mechanisms, interfaces, and semantics. Service-oriented architectures are naturally relevant technology to facilitate such a multi-domain formation through service composition and orchestration. They asserted that it is important to leverage existing research on multi-domain policy integration and the secure-service composition to build a comprehensive policy-based management framework in cloud computing environments.

In our point of view the scenario appears much more complex if we make a binding between the heterogeneity of environments with the possibility of federating them. The concept of federation has always had both political and historical implications: the term refers, in fact, to a type of system organization characterized by a joining of partially "self-governing" entities united by a "central government". In a federation, each self-governing status of the component entities is typically independent and may not be altered by a unilateral decision of the "central government". More specifically, looking at the political philosophy, the federation refers to the form of government or constitutional structure known as federalism and can be considered the opposite of the "unitary state". The components of a federation are in some sense "sovereign" with a certain degree of autonomy from the "central government": this is why a federation can be intended as more than a mere loose alliance of independent entities [7].

Considering the federation perspective in cloud computing environments, new terms are also been coined as Intercloud ("Think of the existing cloud islands merging into a new, interoperable Intercloud where applications can be moved to and operate across multiple platforms..." [8]) or Cross-cloud ("For the benefit of human society and the development of cloud computing, one uniform and interoperable Cross-cloud platform will surely be born in the near future..." [9]). Nowadays, cloud federation is becoming a topic more and more debated within both the scientific and ICT industry worlds [10]. In fact, as discussed in [11], it brings many new business advantages for the enhancement of cloud providers' profit. In such a perspective, new paradigms allowing providers to avoid the limitation of owning only a restricted amount of resources are rising.

Nevertheless, a few works are available in literature related to federation in cloud computing environments. The main reason is that several pending issues concerning security and privacy still have to be addressed, and a fortiori, is not clear what cloud federation actually means and what the involved issues are [12].

Nowadays, the latest trend to federate applications and service oriented architectures (SOAs) over the Internet is represented by the Identity Provider/Service Provider (Id-P/SP) model [13]. Examples are the aforementioned SAML, OpenID [14], Shibboleth [15] and Cardspace [16]. Such solutions, considered alone, do not solve the cloud federation issues. In fact, the federation problem in cloud computing is greater than the one in traditional systems. The main limit of the existing federation solutions is that they are designed for static environments requiring a priori policy agreements, whereas clouds are high-dynamic and heterogeneous environments, which require particular automatic security and policy arrangements. Keeping in mind the cloud federation perspective, several security issues are already picked out. Interoperability in federated heterogeneous cloud environments is faced in [9], in which the authors propose a trust model where the trust is delegated between trustworthy parties, which satisfy certain constrains.

Nevertheless, such works do not fully clarify what it is really meant with the term cloud federation. Basically, it is not fully evaluated when, why, and how a cloud federation should be established and what the impact over the existing infrastructure, the involved architectural issues, and the

security concerns are. Therefore, we think a cloud federation model addressing architectural and security issues, also with implementation practice compliant with existing cloud infrastructures, is strongly needed.

III. Cross-Cloud Federation Analysis: our Reference Scenario

In this Section we try to clarify ideas concerning the general concept of cross-cloud federation. In order to identify requirements and goals, we propose a possible resource provisioning scenario where clouds might benefit of federation advantages. Cloud Computing relies its computational capabilities exploiting the concept of "virtualization". This technology has re-emerged in recent years as a compelling approach of increasing resource utilization and reducing IT services costs. The common theme of all virtualization technologies is hiding the underlying infrastructure by introducing a logical layer between the physical infrastructure and the computational processes. The virtualization is being possible thanks to Virtualization Machine Monitors (VMMs commonly known as "hypervisors"), i.e. processes that run on top of a given hardware platform, control and emulate one or more other computer environments (virtual machines). Each of these virtual machines, in turn, runs its respective "guest" software, typically an operating system, executed as if it is installed on a stand-alone hardware platform.

Private clouds hold their own virtualization infrastructure where several virtual machines are hosted to provide services to their clients. When argument of discussion is cloud federation the skeptics could ask: why should a cloud federate itself with other clouds? In our opinion, the answer is simple: cloud federation brings new business opportunities. In fact, in a scenario of "cross-cloud federation", each cloud operator is able to transparently enlarge its own virtualization resources amount (i.e., increasing the number of instantiable virtual machines and therefore cloud services) asking computing and storage capabilities to other clouds.

According to our analysis, within the above mentioned scenario we distinguish among cloud's client, home cloud and foreign cloud:

- **Cloud's client**. An IT company, organization, university, generic single end-user ranging from desktop to mobile users or a cloud provider using the *aaS supplied by a target "home cloud" according to a pay-per-use model.
- **Home cloud**. A cloud provider which receives *aaS instantiation requests by its clients. Each home cloud for the arrangement, composition, and delivery of such services can use the computing and storage resources of its own virtualization infrastructure along with the resources borrowed by foreign clouds according to a pay-per-use model.
- **Foreign cloud**. A cloud provider which lends its storage and computing resources to home clouds according to a pay-per-use model. More specifically, a foreign cloud reserves part of its own virtualization infrastructure for a home cloud, so that the home cloud can logically count on an elastic virtualization infrastructure whose capabilities are greater than the capabilities of its own physical virtualization infrastructure. Therefore even thought the virtual environments and services of a home cloud are logically placed in its virtualization infrastructure, in reality they can be physically placed in parts of the virtualization infrastructure lent by foreign clouds. A cloud provider could be at the same time both home cloud and/or foreign cloud.

There is not limit to the possible business scenarios which can take place in a cross-cloud federation environment. Such scenarios include, for example, virtualization capability enlargement, resource optimization, provisioning of distributed IaaS, PaaS, and SaaS spread over different clouds, power saving and so on.

In order to better explain the idea of cross-cloud federation, let us consider as reference the "virtualization capability enlargement" scenario depicted in Figure 1. When a home cloud realizes that its virtualization infrastructure has saturated its capabilities, in order to continue providing services to its clients (i.e., other clouds, enterprises, generic end users, etc), it decides to federate itself with foreign clouds A and B. The home cloud, besides hosting virtualization resources inside its own virtualization infrastructure, is also able to hosting virtual machines inside the foreign clouds A and B virtualization infrastructures, enlarging the amount of its available virtualization resources (See Figure 1, bottom part). Therefore, although the virtualization resources rent to the home cloud are physically placed within the virtualization infrastructures of foreign clouds A and B, they are logically considered as resources indeed hosted within the home cloud virtualization infrastructure. Despite the obvious advantages, the implementation of such cross-cloud federation scenario is not at all trivial. The main reason is that clouds are more complicated than traditional systems and the existing federation models are not applicable. In fact, while clouds are typically heterogeneous and dynamic, the existing federation models are designed for static environments where it is needed an a priori agreement among the parties to make up the federation. Keeping in mind the aforementioned scenario, we think cloud federation needs to meet the following requirements: *a) automatism and scalability*, a home cloud, using discovery mechanisms, should be able to pick out the right foreign clouds which satisfies its requirements reacting also to cloud changes; *b) interoperable security*, it is needed the integration of different security technologies, for example, permitting a home cloud to be able to join the federation without changing its security policies. In the "interoperable security" context we identify: *1) SSO authentication*, a home cloud should

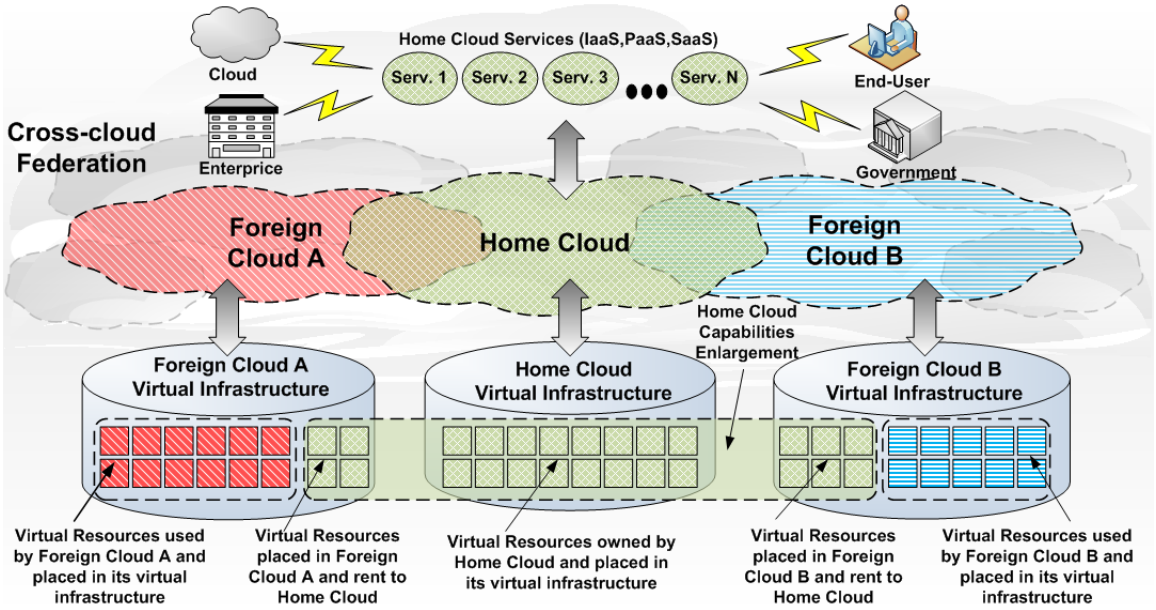

Figure 1. Cross-Cloud Scenario: basic for heterogeneous and federated clouds.

be able to authenticate itself once gaining the access to the resources provided by federated foreign clouds belonging to the same trust context without further identity checks; *2) digital identities and third parties*, each home cloud should be able to authenticate itself with foreign clouds using its digital identity guaranteed by a third party. This latter feature is more challenging because it implies a cloud has to be considered as a subject uniquely identified by some credentials.

IV. Cross-cloud federation: Architectural Overview

In Section III, we described the concept of federation and its bindings with the cloud. In the following, we provide a detailed description of the approach used to address the cross-cloud federation issues. Considering the requirements of automatism, scalability and interoperability previously stated, our solution tries to answer all such issues. Describing the federation process we point out three main different phases: *discovery*, *match-making* and *authentication*. These phases are opportunely explained in the following.

A. The Three-Phase Cross-Cloud Federation Model and the Cross-Cloud Federation Manager

In order to identify the main components constituting a cloud and better explain the federation idea on which our work is based, we are considering the internal architecture of each cloud as the three-layered stack [17] presented schematically in Figure 2.

B. Architectural Overview

Starting from the bottom, we can identify: *Virtual Machine Manager*, *Virtual Infrastructure (VI) Manager* and *Cloud Manager*. VI Manager is a fundamental component of private/hybrid clouds acting as a dynamic orchestrator of Virtual Environments (VEs), which automates VEs setup, deployment and management, regardless of the underlying Virtual Machine Manager layer (i.e., Xen, KVM, or VMware). The Cloud Manager layer is instead able to transform the existing infrastructure into a cloud, providing cloud-like interfaces and higher-level characteristics for security, contextualization and VM disk image management.

In a cloud architecture designed according to the aforementioned three-layered stack, all the cloud components and their respective functions are clearly defined and separated, thus introducing simplicity and efficiency when the cloud middleware has to be modified or new features have to be added. In our work, we exploited such modular characteristics of the layered cloud architecture, and introduced a new component within the Cloud Manager layer (depicted in the top part of Figure 2), named *Cross-Cloud Federation Manager* (CCFM). The CCFM has been conceived for enabling each cloud to perform all the operations needed to pursue the target of the federation establishment.

The cross-cloud scenario we are considering can be seen as an highly dynamic environment: new clouds, offering different available resources and different authentication mechanisms could appear, while others could disappear. Taking into account such dynamism, when a home cloud needs to "lease" external resources from a foreign cloud, the first step the home cloud will perform refers to the

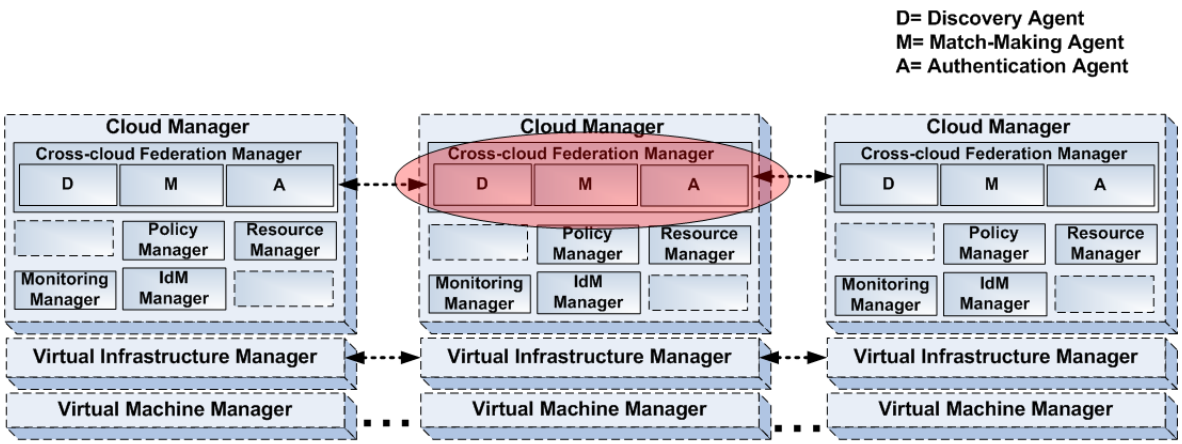

Figure 2. General federated three-tier cloud architectures.

discovery (phase 1) of the foreign cloud, which properly *matches* (phase 2) its requirements (both in terms of available resources and supported authentication mechanisms). Once these two steps have been performed, and the best foreign cloud has been found, in order to establish a secure interaction between the home cloud and the selected foreign cloud, an *authentication* (phase 3) process will begin.

The CCFM module represents the main "actor" in our three-phase federation model. In our design, it consists of three different subcomponents (agents) each addressing a different phase of the federation model:

- The *discovery agent* manages the discovery process among all the available clouds within the dynamic environment. Since its state is pretty flexible and dynamic, the discovery process has to be implemented in a totally distributed fashion: all the discovery agents must communicate exploiting a p2p approach.
- The *match-making agent* accomplishes the task of choosing the more convenient foreign cloud, evaluating all the parameters regarding the QoS, available resources and available authentication mechanisms. By means of specific algorithms, this agent is able to evaluate from all the available (discovered) clouds, the ones that best "fit" the requirements (e.g. the load capacity in terms of resources leasing and the supported authentication methods) of its home cloud.
- The *authentication agent*, cooperating with third parties trusted entities, takes part in the creation of a security context between home and foreign clouds. When the authentication phase begins, the home cloud authentication agent contacts its "peer" on the foreign cloud: the authentication process between such agents (and thus the clouds) will be lead exchanging authentication information in form of meta-data, also involving trusted third parties in the process. The Authentication Agent communicates both with other peers and third parties via web service interfaces.

The accomplishment of the authentication process, carried out by the authentication agents of both home and foreign clouds, leads to the establishment of a secure and direct connection between the related VI Manager Layer of the same clouds. As consequence, the home cloud will be able to instantiate (or migrate) Virtual Resources (VMs) on the Foreign Cloud in a secure environment. The concept of migration can be seen as the opportunity to move the Virtual Machines not only in intra-site domain but also to transfer them on federated inter-site domains. In this case the migration might occur across subnets, among hosts that do not share storage and across administrative boundaries.

Although in Section IV-C we'll describe the three phases needed to pursue the cloud federation, the main scope of this paper refers to the solution of the cloud SSO authentication problem.

In our work, the practical solution to overcome the authentication problem is the introduction the well-known concept of Identity Provider (IdP) along with a new SAML profile (further details are presented in Section V).

C. The Three-Phase Cross-Cloud Federation Process

In this Section, we provide a more detailed description of the three-phase cross-cloud federation model considering the scenario represented in Figure 3. As depicted, such a scenario includes both home clouds and foreign clouds, which are represented according to the three-tier architecture already discussed. More specifically, each cloud platform is composed as follows: The highest stack level includes the CCFM module, which comes with its own agents (discovery, match-making and authentication), instead the underlying layer includes a generic VI Manager. Instead, for our dissertation it is not relevant the specific solution employed at the lowest layer.

We remarked the need of providing a global authentication mechanism exploitable from all the entities belonging to the

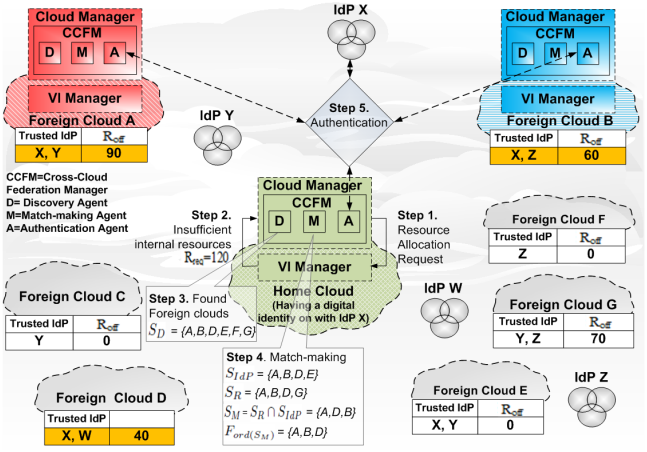

Figure 3. Example of cross-cloud federation establishment.

cloud federation. In Figure 3, together with home clouds and foreign clouds, IdPs are also depicted. An IdP is a provider of digital identity representing a trusted third party, which provides authentication services to its clients. In such scenario we assume each home cloud must have one digital identity at least on one IdP (even though many cloud digital identities may exist on different IdPs), whereas each foreign cloud must be trusted or compliant with one or more IdPs. Before explaining our motivation to the introduction of IdP within our scenario, we provide a description of the three phases needed to achieve the cloud federation.

In the scenario of federation establishment depicted in Figure 3, during the step 1, the home cloud manager layer receives a request for services from its clients and sends a resource allocation request (i.e. virtual machines) to the underlying VI manager layer. In step 2 the home cloud VI manager, evaluating its instantaneous workload, replies to the request notifying it has not enough resources. In step 3 (the discovery phase) the home cloud manager decides to ask for resources to foreign clouds: the resource request is forwarded to the CCFM, which, by means of its discovery agent, will begin the *discovery* process to obtain a list of all the available foreign clouds. The discovery phase can exploit whatever p2p approach to achieve the complete list of cloud providers. Each discovered foreign cloud is associated to a set of meta-data describing several cloud information: the amount and type of the resources available for leasing, the offered SLA level and the supported IdP(s). In this particular example, the agent has found the set of discovered foreign clouds $S_D = \{A, B, C, D, E, F, G\}$.

In step 4 the *match-making* phase begins: the match-making agent of the home cloud selects from the set of discovered foreign clouds S_D the ones, which fits its requirements. The adopted criteria to perform the selection is based on two different evaluation tasks: in the first one, starting from the foreign clouds set S_D, a new subset $S_R = \{A, B, D, G\}$ is obtained considering the foreign clouds better satisfying the home cloud request in terms of resources availability (CPU, RAM, storage) and QoS. In the second evaluation task, starting from the discovered foreign clouds set S_D, the match-making agent selects the subset of foreign clouds $S_{IdP} = \{A, B, D, E\}$, having trusted relationship(s) with the IdP(s) on which the home cloud already has a digital identity. In this example foreign clouds A, B, D, and E are trusted with the IdP X, which provides authentication services to the home cloud guaranteeing for its digital identity. The subsequent operation accomplished by the match-making agent refers to the definition of the set of match-made clouds $S_M = S_R \cap S_{IdP}$.

We now define the metrics R_{req} and $R_{off}(F_i)$ representing respectively a measure of the resources requested by the home cloud, and a measure of the resources offered by the foreign cloud F_i. The value of the metric is obtained evaluating different parameters such as CPU, RAM, storage and QoS for both R_{req} and $R_{off}(F_i)$. In order to identify which foreign clouds fit the home cloud requirements, the match-making agent achieves a list of preferred foreign clouds $F_{ord(S_M)} = \{F_1, F_2, \ldots, F_n\}$ considering the set S_M and ordering its element by the R_{off} value, in a descending order.

Considering the example depicted in Figure 3, $S_M = \{A, D, B\}$ and $F_{ord(S_M)} = \{A, B, D\}$. The match-making agent has to consider the resources provided by the first k foreign clouds of $F_{ord(S_M)}$ to satisfy the condition $R_{req} \leq \sum_{i=1}^{k} R_{off}(F_i), 1 \leq k \leq n$ (in the scenario depicted in Figure 3, we assume $k = 2$ and consequently both foreign clouds A and B will be chosen to establish the federation).

In step 5 (authentication phase), in order to establish a federation with foreign cloud A and B, a cloud SSO *authentication* process has to be started by the home cloud. Such process will involve: the authentication agent of the home cloud, the corresponding peers of the foreign cloud A and B and the IdP X (trusted with A and B, on which the home cloud has a digital identity) where the home cloud performs a SSO log-in. Once the home cloud and foreign cloud A authentication agents establish a trust context, their respective underlying VI manager layers setup a low-level trust context allowing the cross-cloud resource provisioning. Therefore, the home cloud VI manager will be able to instantiate virtual resources on the foreign cloud VI manager. Even if cross-cloud federation has to be established also with foreign cloud D, no further authentication tasks would be needed because foreign cloud D has already a trusted relationship with IdP X.

As can be perceived, the employment of the IdPs presents some advantages well fitting our cross-cloud federation scenario: even though each cloud has its internal security mechanisms, whatever the *foreign cloud* is, regardless of its authentication mechanisms, by means of IdPs a *home cloud* will be able to authenticate itself with other foreign

clouds already having a trust relationship, exploiting the well-known concept of SSO. The resource provisioning in cross-cloud federation may be solved establishing trust relationships between the clouds using several IdPs containing the credentials of the cloud asking for resources. Section V better describes the steps involved in phase 5, pointing out the technologies employed to implement the authentication and the set of information exchanged between the involved entities. The same Section describes our new SAML profile designed to accomplish the cloud SSO authentication in a federated scenario.

V. The Authentication Phase Using SAML

In this Section, after a brief description of the SAML standard, we focus on the authentication phase (step 5 of Figure 3) of our three-phase cross-cloud federation model performed by the Authentication Agent. More specifically, using the SAML technology we propose a new *Cross-Cloud Authentication Agent SSO Profile*, which describes the messages exchanging flow between a home cloud, foreign clouds and IdPs during the establishment of a trust context.

A. SAML Technology Overview

SAML is an XML-based standard for exchanging authentication and authorization assertions between security domains, more specifically, between an Identity Provider (IdP) (a producer of assertions) and a generic Service Provider (SP) (a consumer of assertions). SAML consists of: a subject, a person or a software/hardware entity that assumes a particular digital identity and interacts with an online application, composed of several heterogeneous systems; a SP or relying party, a system, or administrative domain, that relies on information supplied to it by the Identity Provider; an IdP or asserting party, a system, or administrative domain, that asserts information about a subject. In literature, such a model is also referred as IdP/SP.

The aim of SAML is enable a principal to perform SSO. This means a principal, by means of its IdP, must be able to authenticate itself once gaining the access to several trusted service providers which might use also different security technologies. SAML assumes the principal has enrolled in at least one identity provider offering SSO authentication services. The main advantage of SAML is it does not care how authentication services are implemented. In fact, the whole SAML authentication process of a subject on a service provider is performed by means of a set of messages exchanging SAML security assertions. Service providers, in order to authenticate a principal, merely relies on the assertion sent by the trusted IdP (on which the principal is enrolled).

SAML combines four key concepts: assertion, binding, protocol and profile. Assertion consists of a package of information that supplies one or more statements (i.e. authentication, attribute, and authorization decision) made by the IdP. Authentication statement is perhaps the most important meaning the IdP has authenticated a subject at a certain time. A Protocol (i.e. Authentication Request, Assertion Query and Request, Artifact Resolution, etc) defines how subject, service provider, and IdP might obtain assertions. More specifically, it describes how assertions and SAML elements are packaged within SAML request and response elements. A SAML binding (i.e. SAML SOAP, Reverse SOAP (PAOS), HTTP Redirect (GET), HTTP POST Binding, etc) is a mapping of a SAML protocol message over standard messaging formats and/or communications protocols. For example, the SAML SOAP binding specifies how a SAML message is encapsulated in a SOAP envelope. A profile (i.e. Web Browser SSO, Enhanced Client or Proxy (ECP), Single Logout, Attribute Profiles, etc) is a technical description of how a particular combination of assertions, protocols, and bindings defines how SAML can be used to address particular scenarios.

B. The SAML Cross-Cloud Authentication Agent SSO (CCAA-SSO) Profile

The authentication agent has been designed both to manage the digital identity of the home cloud and to perform authentication tasks sending/receiving authentication requests to/from foreign clouds, interacting with their respective peer modules. More specifically, the authentication agent does not directly manages the digital identity of the cloud, but uses one or more trusted IdPs acting as guarantor when the agent likes to authenticate the home cloud with other foreign clouds during the federation establishment.

To address the cloud SSO authentication problem, during the cross-cloud federation establishment we developed a new SAML profile named Cross-Cloud Authentication Agent SSO (CCAA-SSO). This profile was designed to enable a home cloud to perform SSO authentication on several foreign clouds both having a trusted relationship with the home cloud's IdP and regardless their security mechanisms. Such an authentication process is fundamental for the subsequent establishment of a secure interaction between the home cloud VI manager and one or more foreign cloud VI manager(s). Once the secure interaction has been established the home cloud is able to gain the access to the required resources offered by the foreign cloud. More specifically, the profile defines the messages exchange flow between the home cloud, the foreign clouds, and the IdP, solving the cloud SSO authentication problem. The implementation of the profile has been accomplished using and extending the java libraries of the OpenSAML project [18] for both the authentication agents and the IdP. In order to accomplish such tasks, the agent has been developed exposing a web service interface using the SOAP [19] technology, but nothing prevents the adoption of other web service technologies such as REST, JAX-RPC, or XML-RPC.

In a CCAA-SSO profile use case, both the home cloud

and the foreign cloud, by means of their own Authentication Agents, represent respectively the subject and the relying party, whereas the IdP acts as the third party asserting to a foreign cloud the trustiness of the home cloud identity. The CCAA-SSO profile has been designed as a combination of the following SAML elements: an assertion including an authentication statement, a request-response protocol, and a SAML SOAP Binding.

Considering the scenario already pointed out in Section IV-C, in the following we describe the authentication process previously marked as phase 5 keeping in mind our SAML CCAA-SSO profile considering a VI Manager. In Figure 4 is shown the flow of messages exchanged between the home cloud, the foreign cloud A and the IdP X, putting aside for the time being foreign cloud B. More specifically, inside each cloud both Authentication Agent and the VI Manager are involved in the process. In step 5.1 the Authentication

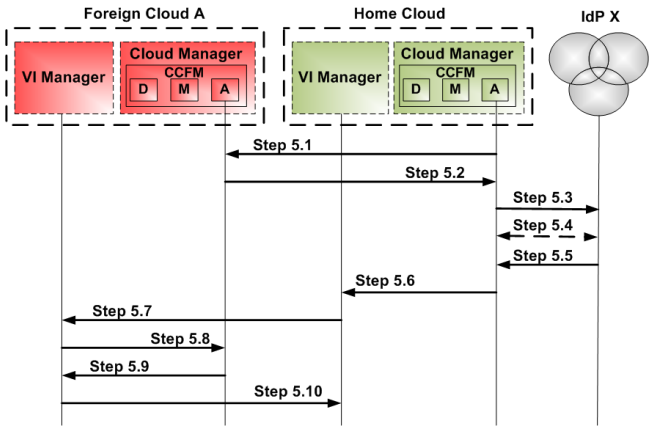

Figure 4. Sequence diagram describing the steps of the CCAA-SSO profile during the authentication of the home cloud with the foreign cloud A by means of the IdP X.

Agent, on behalf of the home cloud manager, forwards to the corresponding peer of the foreign cloud A a SOAP request for a set of virtual resources by means of a XML document. In step 5.2 the Authentication Agent of the foreign cloud A responds to the home cloud with a SAML authentication request enveloped in a SOAP message. In step 5.3 the Authentication Agent of the home cloud unpacks the authentication request received at step 5.2 and forwards it via SAML/SOAP message to the IdP X, making a SSO request. As a valid trust context does not exist, in step 5.4 the IdP X authenticates the home cloud using a given security technology (the independence from the security technology used by each cloud is accomplished). In step 5.5, as the home cloud identity is verified, the IdP X responds to the authentication request by means of a SAML/SOAP response message, signing it with its private key. In step 5.6 the Authentication Agent of the home cloud unpacks the authentication assertion received in step 5.5 and forwards it to the underlying VI Manager. In step 5.7 the VI manager of the home cloud sends the authentication assertion via SAML/SOAP to the corresponding peer of the foreign cloud A. In step 5.8 the VI manager of the foreign cloud B forwards the received authentication statement to its authentication agent, which proves its correctness verifying the digital sign using the public key of the IdP X (see step 5.5). In step 5.9 the VI manager of the foreign cloud B receives a notification about the authentication assertion validity and authenticate the home cloud VI Manager establishing a secure interaction. In step 5.10 the VI manager of the foreign cloud provides the resources requested by the home cloud at step 5.1

The authentication process of the home cloud with the foreign cloud B is analogous to the one already described for foreign cloud A, with one important difference: since the home cloud has already performed the authentication on the IdP X in the step 5.4, no further authentication is needed because a trust context already exists (the SSO is thus accomplished). Therefore, the SAML CCAA-SSO profile combines both security and flexibility ensuring cloud SSO authentication in cross-cloud federation environments between clouds representing a possible solution for secure federated cloud interactions.

In the Section VII the same sequence of steps involved in the CCAA-SSO profile will be considered again, in a new scenario where a new Virtual Infastructure Manager named CLEVER will be introduced: thanks to its features, the CLEVER middleware well fits the requirement of dynamic resource sharing that is a pillar of a federated cloud environment. All the details regarding this middleware will be discussed in the following Section (VI).

VI. CLEVER: A VIRTUAL INFRASTRUCTURE MANAGER

In this Section we will focus on the description of CLEVER, a VI Manager which aims to provide *Virtual Infrastructure Management* services and suitable interfaces at the *High-level Management* layer to enable the integration of high-level features such as Public Cloud Interfaces, Contextualization, Security and Dynamic Resources provisioning. After a brief description of both its architecture and its main features, the way by means it is possible to interconnect different CLEVER domains will be discussed.

A. CLEVER: an Architectural Overview

Looking at the existing middleware implementations, which act as *High-level Cloud Manager* [20], [21], it can be said that their architecture lacks modularity: it could be a difficult task to change these cloud middleware for integrating new features or modifying the existing ones. CLEVER instead intends granting an higher scalability, modularity and flexibility exploiting the plug-ins concept. This means that other features can be easily added to the middleware just introducing new plug-ins or modules within its architecture without upsetting the organization.

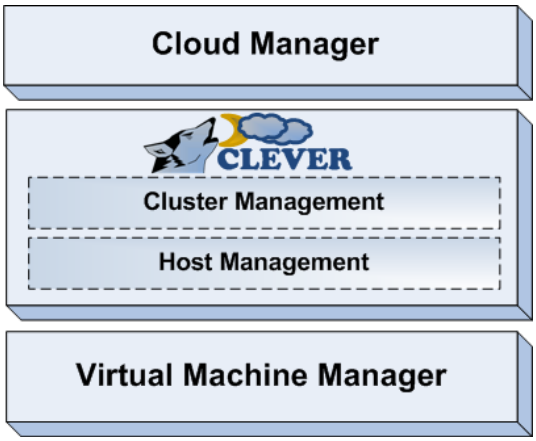

Figure 5. How the VI Manager layer is implemented in CLEVER: the cluster management and the host management

Furthermore, analysing the current existing middleware [22], [23], which deal with the *Virtual Infrastructure Management*, some new features could be added within their implementation in order to achieve a system able to grant high modularity, scalability and fault tolerance. The main idea on which CLEVER is based, finds in the terms flexibility and scalability its key-concepts, leading to an architecture designed to satisfy the following requirements: 1) *persistent communication* among middleware entities; 2) *transparency* respect to "user" requests; 3) *fault tolerance* against crashes of both physical hosts and single software modules; 4) heavy *modular design* (e.g. monitoring operations, managing of hypervisor and managing of VEs images will be performed by specific plug-ins, according to different OS, different hypervisor technologies, etc); 5) *scalability* and *simplicity* when new resources have to be added, organized in new hosts (within the same cluster) or in new clusters (within the same cloud); 6) automatic and optimal *system workload balancing* by means of dynamic VEs allocation and live VEs migration. The typical scenario where CLEVER could be deployed consists of a set of physical hardware resources (i.e., a cluster) where VEs are dynamically created and executed on the hosts considering their workload, data location and several other parameters. The basic operations our middleware should perform refer to: 1) Monitoring the VEs behavior and performance, in terms of CPU, memory and storage usage; 2) Managing the VEs, providing functions to destroy, shut-down, migrate and network setting; 3) Managing the VEs images, i.e., images discovery, file transfer and uploading.

Considering the concepts stated in [4] and looking at Figure 5, such features, usually implemented in the *Virtual Infrastructure Management* layer, can be further analyzed and arranged on two different sub-layers: *Host Management* (lower) and *Cluster Management* (higher).

Grounding the design of the middleware on such logical subdivision and taking into account the satisfaction of all the above mentioned requirements, the simplest approach to design our middleware is based on the architecture schema depicted in Fig. 6, which shows a cluster of n nodes (also an interconnection of clusters could be analyzed) each containing a *host level* management module (Host Manager). A single node may also include a *cluster level* management module (Cluster Manager). All these entities interact exchanging information by means of the *Communication System* based on the Extensible Messaging and Presence Protocol (XMPP) [24]. In particular, the main entities of CLEVER, communicates each other "talking" and exchanging messages within the same XMPP room, like in a traditional chat. As the Figure 6 shows, a CM coordinates the HMs of the whole cluster sending XMPP messages to them by means of a multi-user-chat.

Finally, the set of data necessary to enable the middleware functioning is stored within a specific *Database* deployed in a distributed fashion.

Figure 6 shows the main components of the CLEVER architecture, which can be split into two logical categories: software agents (typical of the architecture itself) and the tools they exploit. To the former set belong both *Host Manager* and *Cluster Manager*:

- **Cluster Manager (CM)** acts as an interface between the clients (software entities, which can exploit the cloud) and the HM agents. CM receives commands from the clients, performs operations on the HM agents (or on the database) and finally sends information to the clients. It also performs the management of VE images (uploading, discover, etc.) and the monitoring of the overall state of the cluster (resource usage, VEs state, etc.). Following our idea, at least one CM has to be deployed on each cluster but, in order to ensure higher fault tolerance, many of them should exist. A master CM will exist in active state while the other ones will remain in a monitoring state, although client messages are listened whatever operation is performed.
- **Host manager (HM)** performs the operations needed to monitor the physical resources and the instantiated VEs; moreover, it runs the VEs on the physical hosts (downloading the VE image) and performs the migration of VEs (more precisely, it performs the low level aspects of this operation). To carry out these functions it must communicate with the hypervisor, hosts' OS and distributed file-system on which the VE images are stored. This interaction must be performed using a plug-ins paradigm.

Regarding the tools such middleware components exploit, we can identify the *Distributed Database* and the *XMPP Server*:

- **Distributed Database** is merely the database containing the overall set of information related to the

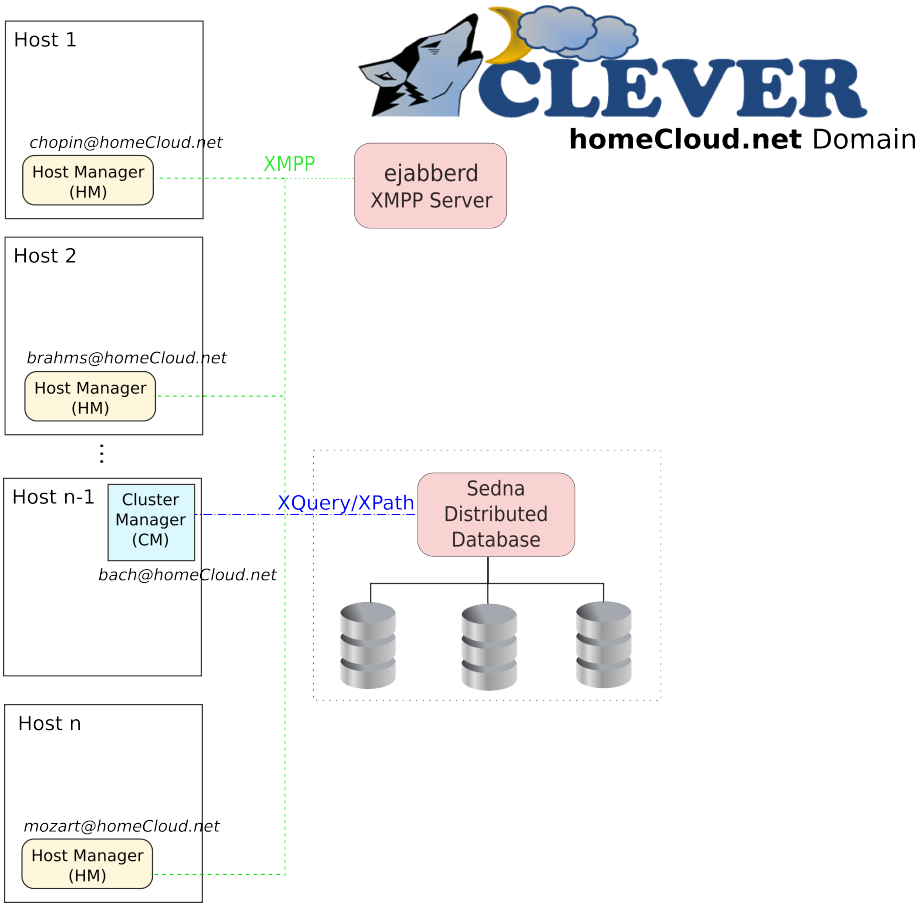

Figure 6. CLEVER reference scenario.

middleware (e.g. the current state of the VEs or data related to the connection existing on the Communication System). Since the database could represent a centralized point of failure, it has to be developed according to a well structured approach, for enabling fault tolerance features. The best way to achieve such features consists of using a Distributed Database. In the current CLEVER implementation, the database is based on the sedna native XML database system [25]. Sedna provides a full range of core database services (e.g., persistent storage, ACID transactions, security, indices, hot backup). Flexible XML processing facilities include W3C XQuery implementation, tight integration of XQuery with full-text search facilities and a node-level update language.

- **XMPP Server** is the "channel" used to enable the interaction among the middleware components. In order to grant the satisfaction of our requirements, it is able to offer: decentralization (i.e., no central master server should exist: such capability in native on the XMPP) in a way similar to a p2p communication system for granting fault-tolerance and scalability when new hosts are added in the infrastructure; flexibility to maintain system interoperability; security based on the use of channel encryption: since the XMPP Server also could exploit the distributed database to work, the solution enables an high fault tolerance level and allows system status recovery if a crash occurs. All these features are guaranteed in the current implementation by means of the employment of the Ejabberd XMPP server [24]. Ejabberd is a Jabber/XMPP instant messaging server, licensed under GPLv2 (Free and Open Source), written in Erlang/OTP. Among other features, ejabberd is cross-platform, fault-tolerant, clusterable and modular.

B. How CLEVER Supports Interdomain Communication

Even though the XMPP uses a client/server model there is not a central authoritative server. As anyone may run its own XMPP server on its own domain, it is the interconnection among these servers which makes up the XMPP network. Every user on the network has a unique Jabber ID (JID). To avoid requiring a central server to maintain a list of IDs, the JID is structured like an e-mail address with a user name and a domain name for the server where that

user resides, separated by an at sign (@). For example, considering the CLEVER scenario a CM could be identified by a JID *bach@homeCloud.net*, whereas a HM could be identified by a JID *liszt@foreignCloudA.net*: *bach* and *liszt* respectively represent the host names of the CM and the HM, instead *homeCloud.net* and *foreignCloudA.net* represent the domains of the cloud resources.

Let us suppose that *bach@homeCloud.net* wants to communicate with *liszt@foreignCloudA.net*, *bach* and *liszt*, each respectively, have accounts on the *homeCloud.net* and *foreignCloudA.net* XMPP servers. When *bach* wants to start the communication, a sequence of events is triggered:

1) *bach* sends its message to the *homeCloud.net* server
2) The *homeCloud.net* server opens a connection to the *foreignCloudA.net* server.
   a) If *homeCloud.net* server has successfully performed the authentication with *foreignCloudA.net* server, the messagge is forwarded.
   b) If *homeCloud.net* server has not successfully performed the authentication on *foreignCloudA.net* server, the message is dropped.
3) If 2a is verified, the *foreignCloudA.net* server checks to see if *liszt* is currently connected. If not, the message is stored for later delivery.
4) If *liszt* is online, the *foreignCloudA.net* server delivers the message to *liszt*.

A CLEVER cluster includes a set of HMs, orchestrated by a CM, all acting on a specific domain and connected to the same XMPP room. Each HM is deployed in a physical host and is responsible to manage its computing and storage resources according to the commands given by the CM. The idea of federation in CLEVER environments is founded on the concept that if a CLEVER cluster on a domain needs of external resources of other CLEVER clusters, acting on different domains, a sharing of resources can be accomplished, so that the resources belonging to a domain can be logically included in another domain. Within CLEVER this is straightforward by means of the built-in XMPP features.

Considering the aforementioned domains *homeCloud.net* and *foreignCloudA.net*, in scenarios without federation, they respectively include a different XMPP rooms (i.e., *cleverRoom@homeCloud.net* and *cleverRoom@foreignCloudA.net*) on which a single CM, responsible for the administration of the domain, communicates with several HMs, typically placed within the physical cluster of the CLEVER domain. Instead, considering a federated scenario among the two domains, if the CM *bach* of the *homeCloud.net* domain needs of external resources, it could invite within its *cleverRoom@homeCloud.net* room one or more HMs of the *foreignCloudA.net* domain. As previously stated, in order to accomplish such a task a trust relationship between the *homeCloud.net* and the *foreignCloudA.net* XMPP server has to be established. Such a concept will be better clarified in Section VII by means of a concrete use case.

VII. The CCAA-SSO Profile Applied to CLEVER

In Section V-B we described the authentication process in a generic cloud environment, pointing out the sequence of step involved in the CCAA-SSO profile. In the following, keeping in mind the aforementioned description of the CLEVER cloud middleware, we aim to discuss how the CCAA-SSO profile may be used in a cloud scenario where CLEVER acts as VI Manager.

This Section will provide more details about the XML documents exchanged among the actors of the profile and will clarify how the resources may be shared among different CLEVER domains exploiting the features offered by the XMPP. In order to describe the process, the same sequence of step already analyzed in Section V-B will be considered. In this description, for simplicity, instead of naming the steps as 5.1-5.10, the notation 1-10 will be employed.

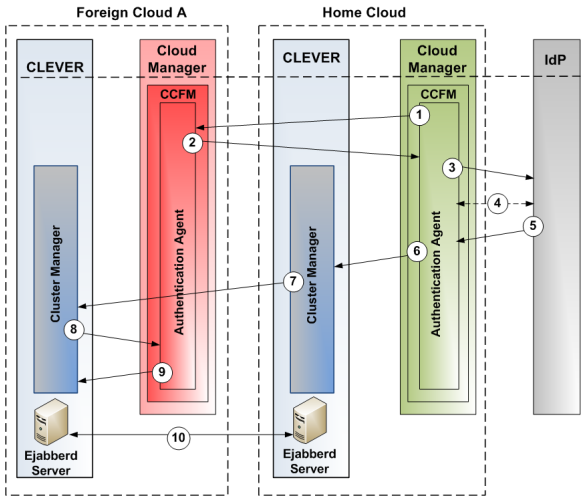

Figure 7. Sequence diagram describing the steps of the CCAA-SSO profile during the authentication between the CLEVER home cloud and the CLEVER foreign cloud A by means of the IdP X.

When the CM of the CLEVER site identified from the domain *homeCloud.net* realizes it has not enough resources to satisfy the stipulated SLA, it tries to ask further resources to an external cloud. These resources are managed from the CM exchanging XMPP messages on the room *cleverRoom@homeCloud.net*, in order to orchestrate the available physical resources (each physical host is managed by an HM).

Supposing that the discovery phase has been accomplished from the *homeCloud.net* and the cloud selected for the federation is *foreignCloudA.net*, the authentication phase can begin according to the steps already

reported in the Section V-B. In this CLEVER scenario, more specifically, during the step 1 the Authentication Agent, on behalf of the home cloud manager, forwards to the corresponding peer of the foreign cloud A a SOAP request (by means of a XML document) for a set of HMs that should join its *cleverRoom@homeCloud.net* to increase the available physical resources. In the SOAP request message reported below, such document is embedded inside the <ResourceType> element and is not depicted for briefness.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/
    envelope/">
   <S:Header/>
   <S:Body>
      <ns2:AA-ForeignCloud-A-ResReq xmlns:ns2="https://
          cloudA.net/SAML2/">
         <ResourceType>"XML resource description
             document"</ResourceType>
      </ns2:AA-ForeignCloud-A-ResReq>
   </S:Body>
</S:Envelope>
```

In step 2 the Authentication Agent of the foreign cloud A responds to the home cloud with a SAML authentication request containing an authentication query. Considering the underlying SAML/SOAP response, the authentication request is provided by means of the element <samlp:AuthnRequest...>.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/
    envelope/">
   <S:Body>
     <ns2:AA-ForeignCloud-A-ResReqResponse xmlns:ns2="
         http://webservices/">
       <return>
        <samlp:AuthnRequest xmlns:samlp="urn:oasis:names
            :tc:SAML:2.0:protocol" xmlns:saml="urn:
            oasis:names:tc:SAML:2.0:assertion" ID="cba2
            " Version="2.0" IssueInstant="2010-11-12T17
            :23:32Z" AssertionConsumerServiceIndex="0">
          <saml:Issuer>https://cloudA.net/SAML2</saml:
              Issuer>
          <samlp:NameIDPolicy
           AllowCreate="true"
           Format="urn:oasis:names:tc:SAML:2.0:nameid-
               format:transient"/>
        </samlp:AuthnRequest>
       </return>
     </ns2:AA-ForeignCloud-A-ResReqResponse>
   </S:Body>
</S:Envelope>
```

In step 3 the Authentication Agent of the home cloud unpacks the authentication request received at step 2 and forwards it via SAML/SOAP to the IdP X, making a SSO request. Since a valid trust context does not exist, in step 4 the IdP X authenticates the home cloud using a given security technology (the independence from the security technology used by each cloud is accomplished). In step 5, since the home cloud identity is verified, the IdP X responds to the authentication request by means of the following SAML/SOAP response, identified by the element <samlp:Response...>. Such element contains an assertion (see element <saml:Assertion...>) with an authentication statement (see element <saml:AuthnStatement...>) and has been signed by the IdP X using its private key (see elements <saml:Issuer> and <ds:Signature xmlns:ds="http//www.w3.org/2000/09/xmldsig#" >).

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/
    envelope/">
<S:Body>
<ns2:IdpX-SSO-ServiceResponse xmlns:ns2="http://
    webservices/">
<return>
   <samlp:Response xmlns:samlp="urn:oasis:names:tc:SAML
       :2.0:protocol" xmlns:saml="urn:oasis:names:tc:
       SAML:2.0:assertion" ID="62af" InResponseTo="cba2"
        Version="2.0" IssueInstant="2010-11-12T17:23:34Z
       " Destination="https://cloudA.net/SAML2/SSO/SOAP
       ">
  <saml:Issuer>https://idpx.net/SAML2</saml:Issuer>
  <samlp:Status>
  <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:
      status:Success"/>
  </samlp:Status>
  <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML
      :2.0:assertion" ID="5c4e" Version="2.0"
      IssueInstant="2010-01-12T18:35:23Z">
    <saml:Issuer>https://idpx.net/SAML2</saml:Issuer>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/
        xmldsig#">
     mgQpzczIazNLSIr8qp7mt0C8jWLBrRsIChVGDML44
     tfZDPCOZfGfbWNBy97ODoEvTtptJtpjp9NN
     JTSweVTofRcy8tHrvLuJnLmMmDbE50KsRoo+yA==
        z6h5g2KOdBkZS7g9w0TJFKlI/OJUOhyodpRr
     8XY9+h/4euIVxg5vXuD6PldBqWgKYtY84+910IP7TXQJS/
        cblOCIf2TdMo55vR0QGDYdBt2yRXd1
     wCUO93dtaSAF6WVid55JE4oraYFEFMfOmgQpzczIazNLSI
     r8qp7mt0C8jWLBrRsIChVGDML44tfZ
     hdttW0jOIazNLSIr8qp7mt0C8jWLBrRsIChVGDML4s+
        xEyyN4hrCEvz2hIcLYA5Q4B1HTKryMCw5
     PIJt0eaTeMicjAyrN+iynUjpW2uAgCvPYHbk4Le/i
    </ds:Signature>
    <saml:Subject>
      <saml:NameID Format="urn:oasis:names:tc:SAML:2.0:
          nameid-format:transient">
        5a42edc7-6439-4de9-12d2-836a74df279c
      </saml:NameID>
      <saml:SubjectConfirmation Method="urn:oasis:names:
          tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData InResponseTo="dfa6"
            Recipient="https://cloudA.net/SAML2/SSO/
            SOAP" NotOnOrAfter="2010-11-12T17:24:34Z"/>
      </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Conditions NotBefore="2010-11-12T17:23:34Z"
        NotOnOrAfter="2010-11-12T17:24:34Z">
      <saml:AudienceRestriction>
        <saml:Audience>https://cloudA.net/SAML2</saml:
            Audience>
      </saml:AudienceRestriction>
    </saml:Conditions>
    <saml:AuthnStatement AuthnInstant="2010-11-12T17
        :24:50Z" SessionIndex="21a4">
      <saml:AuthnContext>
        <saml:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:
              PasswordProtectedTransport
        </saml:AuthnContextClassRef>
      </saml:AuthnContext>
    </saml:AuthnStatement>
  </saml:Assertion>
 </samlp:Response>
</return>
</ns2:IdpX-SSO-ServiceResponse>
</S:Body>
</S:Envelope>
```

In step 6 the Authentication Agent of the home cloud

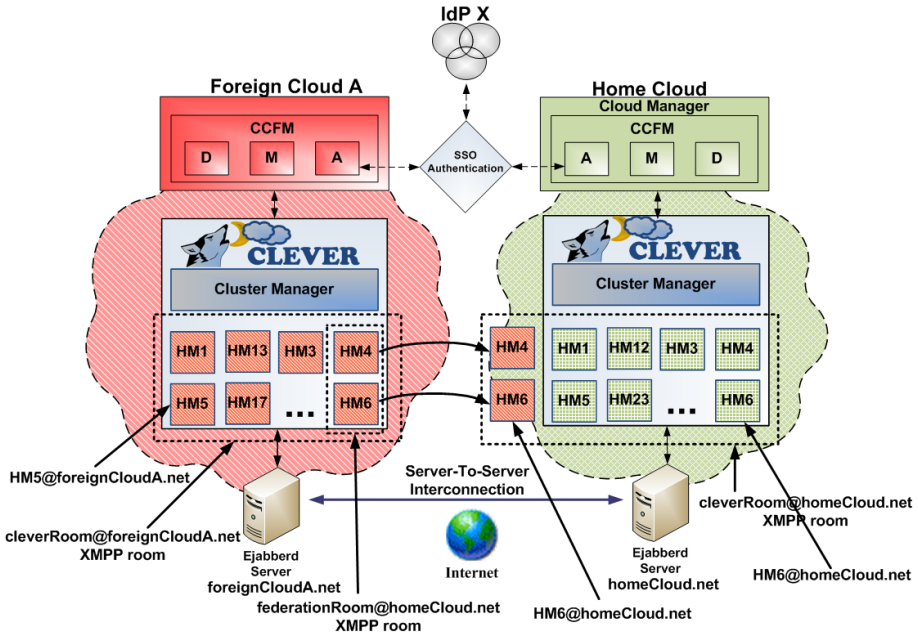

Figure 8. Example of resource sharing among a Home Cloud and a ForeignCloud in a federated scenario.

unpacks the authentication assertion received in step 5 and forwards it to the underlying CLEVER middleware where it is captured from the CM. This latter, in step 7, sends the authentication assertion via SAML/SOAP to the corresponding peer of the foreign cloud A. In step 8 the CLEVER CM of the foreign cloud B forwards the received authentication statement to its authentication agent, which proves its correctness verifying the digital sign using the public key of the IdP X (see step 5).

The Sequence of steps we reported above is represented on the Figure 8. More specifically, the authentication process based on SAML, the IdP and the Authentication Agent is represented on the top part of the Figure by means of the relation *SSO Authentication* which involves these three entities of the scenario.

In step 9 the CLEVER of the foreign cloud B receives a notification about the authentication assertion validity and authenticates the CLEVER home cloud establishing a secure interaction. This interaction leads to the establishment of a server-to-server connection among the homeCloud.net and foreignCloudA.net XMPP servers and therefore an interconnection among the considered CLEVER domains. This operation is represented in the Figure 8 with the double arrow joining the two Ejabberd Server of the two different domains homeCloud.net and foreignCloudA.net (Server-To-Server Interconnection).

In the step 10, the set of resources requested by the home cloud is allocated within the foreign cloud domain: as showed in Figure 8 a temporary XMPP room *federationRoom@foreignCloudA.net*, including two HMs (HM4 and HM6), is created. Furthermore, these HMs will be also included in the *cleverRoom@foreignCloudA.net* room, marked as "rented" as XMPP presence status. This allows the CLEVER CM of the foreign cloud to passively monitor the performance of the rented cluster resources (i.e., the rented HMs) although these are used by the home cloud.

In this way, the CLEVER CM of the foreign cloud is able to rent the HMs included within the *federationRoom@foreignCloudA.net* to the home cloud. As depicted in the Figure, the CLEVER CM of the home cloud can invite the HMs of the *federationRoom@foreignCloudA.net* room to join its *cleverRoom@homeCloud.net* room, so that it can enlarge its available resources for the instantiation of virtual machines.

VIII. Conclusions and Future Works

In this paper we focus on cross-cloud environments, debating the way in which different clouds acting on different administrative domains can establish federation relationships in order to lend and borrow virtualization resources. An approach for the federation establishment consisting of three phases (Discovery, Match-Making, and Authentication) was introduced, and keeping in mind a generic three-tier cloud architecture (from the bottom: VM Manager, VI Manager, and Cloud Manager), a Cross-Cloud Federation Manager (CCFM) responsible for the accomplishment of the aforementioned three phases has been designed and described.

More specifically, the designed CCFM consists of three software agents: Discovery, Match-Making, and Authentication, each one responsible for the accomplishment of a phase during the federation establishment process. In this

paper, we particularly focused on the Authentication Agent, designing a SAML CCAA-SSO profile for generic three-tier cloud architecture.

In our testbed, a use case of the application of the designed SAML CCAA-SSO profile has been performed using cloud platforms consisting on the CLEVER VI Manager and on an implementation of the Authentication Agent of the CCFM module. Details about the sequence of SAML messages exchanged between the involved cloud and the IdP have been provided also by means of several example of XML document captured during the authentication establishment.

Finally, a discussion on the way in which federated CLEVER-based clouds are able, after authentication, to lend and borrow virtualization resources (i.e., in the CLEVER terminology HMs) is provided, also discussing what it happens behind the scenes from the point of view of CLEVER.

In future works, as in a cross-cloud federated environment including hundreds of clouds, considering only one IdP is too much restrictive, we plan to study scenarios including different distributed IdPs also having trustiness relationships each other. In this way we will aim to evaluate the amount of authentications and IdP enrollments needed, either employing real testbeds or by means of a simulated environment, including hundreds of clouds dynamically joining and leaving federations.

REFERENCES

[1] T. Bittman, "The evolution of the cloud computing market," *Gartner Blog Network, http://blogs.gartner.com*, November 2008.

[2] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Three-phase cross-cloud federation model: The cloud sso authentication," *Advances in Future Internet, International Conference on*, vol. 0, pp. 94–101, 2010.

[3] SAML V2.0 Technical Overview, OASIS, http://www.oasis-open.org/committees/ download.php/11511/sstc-saml-tech-overview-2.0-draft-10.pdf.

[4] F. Tusa, M. Paone, M. Villari, and A. Puliafito., "CLEVER: A CLoud-Enabled Virtual EnviRonment," in *15th IEEE Symposium on Computers and CommunicationsS Computing and Communications, 2010. ISCC '10. Riccione*, June 2010.

[5] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE '08*, pp. 1–10, 2008.

[6] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security and Privacy*, vol. 8, pp. 24–31, 2010.

[7] Forum of Federations: http://www.forumfed.org/en/index.php.

[8] Sun Microsystems, Take your business to a Higher Level - Sun cloud computing technology scales your infrastructure to take advantage of new business opportunities, guide, April 2009.

[9] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," in *Cloud Computing*, pp. 69–79, November 2009.

[10] R. Ranjan and R. Buyya, "Decentralized overlay for federation of enterprise clouds," Handbook of Research on Scalable Computing Technologies, 2010. pages: 191-217.

[11] Goiri, J. Guitart, and J. Torres, "Characterizing cloud federation for enhancing providers' profit," *Cloud Computing, IEEE International Conference on*, vol. 0, pp. 123–130, 2010.

[12] N. Leavitt, "Is cloud computing really ready for prime time?," *Computer*, pp. 15–20, January 2009.

[13] Liberty Alleance Project, http://projectliberty.org.

[14] OpenID Authentication 2.0, OpenID Foundation, http://openid.net/specs/openid-attribute-exchange-2_0.html.

[15] The Shibboleth, a Software of the Internet2 Initiative: http://shibboleth.internet2.edu/.

[16] Microsoft Windows Cardspace, http://netfx3.com/content/WindowsCardspaceHome.aspx.

[17] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *Internet Computing, IEEE*, vol. 13, pp. 14–22, September 2009.

[18] OpenSAML, "Open source libraries in Java and C++ providing core message, binding, and profile classes for implementing applications based on SAML 1.0, 1.1, and 2.0", http://saml.xml.org/internet2-opensaml.

[19] "Web services security: Soap message security 1.0, oasis, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf."

[20] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows," in *SWBES 2008, Indianapolis*, December 2008.

[21] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-Source Cloud-Computing System," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pp. 124–131, May 2009.

[22] OpenQRM, "the next generation, open-source Data-center management platform", http://www.openqrm.com/.

[23] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," in *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*, pp. 59–68, June 2009.

[24] Ejabberd, The Erlang Jabber/XMPP Daemon Community: http://www.ejabberd.im/.

[25] Sedna, Native XML Database System: http://modis.ispras.ru/sedna/.