A Diversified Set of Security Features for XMPP Communication Systems Useful in Cloud Computing Federation

Antonio Celesti, Massimo Villari, and Antonio Puliafito
DICIEAMA, University of Messina
Contrada di Dio, S. Agata, 98166 Messina, Italy.
e-mail: {acelesti, mvillari, apuliafito}@unime.it

***Abstract*—Nowadays, in the panorama of Cloud Computing, finding a right compromise between interactivity and security is not trivial at all. Currently, most of Cloud providers base their communication systems on the web service technology. The problem is that both Cloud architectures and services have started as simple but they are becoming increasingly complex. Consequently, web services are often inappropriate. Recently, many operators in both academia and industry are evaluating the eXtensible Messaging and Presence Protocol for the implementation of Cloud communication systems. In fact, the XMPP offers many advantages in term of real-time capabilities, efficient data distribution, service discovery, and inter-domain communication compared to web service technologies. Nevertheless, the protocol lacks of native security features. In this paper, we explore such security issues, discussing how they can be mitigated using both SAML/SASL Single Sign-On (SSO) and XEP 0027.**

Keywords-Cloud computing, federation, security, XMPP, SSO authentication, data encryption, digital signature.

I. Introduction

Nowadays, Cloud providers and their services are becoming more and more complex. Until now, the trend for Cloud Computing has been to base the communication systems on well-known web services such as the Representational State Transfer (REST) and the Simple Object Access Protocol (SOAP). This model has succeeded so far, however, due to the increasingly degree of complexity that Cloud architectures need for fulfilling the new emerging business scenarios, the achievement of both interactivity and security is not trivial at all. In fact, both REST and SOAP web services present the following disadvantages: they are based on request/response patterns, they do not provide any native asynchronous interaction, their polling does not scale and it is not real-time, they require a two-way data exchange. Consequently, web services make complicated *i)* the presence (availability) and discovery of software modules and services; *ii)* many-to-many distribution patterns; *iii)* asynchronous and multi-step calls to remote services; *iv)* federation with third-party providers and services especially behind firewalls.

For these reasons, most operators in both academia and industry fields are looking at alternative communication systems for Cloud providers. To this regards, a valuable solution consists in adopting the Extensible Messaging and Presence Protocol (XMPP), i.e., an open-standard communications protocol for message-oriented middleware based on the XML (Extensible Markup Language). On one hand, the XMPP is able to overcome the disadvantages of web services in terms of performance, but on the other hand it lacks of native security features for addressing the new emerging Cloud computing scenarios.

In this paper, we discuss how the XMPP can be adopted for the development of secure Cloud communication systems. In particular, we combine and generalize the assumptions made in our previous works respectively regarding how to secure inter-domain federation [1] and how to secure inter-module communication [2] with the objective to provider to the reader a guideline.

The US Department of NIST, is actively working for accelerating Standards to foster the Adoption of Cloud Computing [3]. *Interoperability, Portability and Security* are the main objectives to achieve. Considering the intrinsic nature of the XMPP, the first and the second objective can be easily achieved [4], instead the third one deserves further assumptions. More specifically, to achieve secure federation-enabled Cloud architectures over the XMPP, we will discuss how to carry out Single Sign On (SSO) authentication between providers belonging to different administrative domains, data integrity, confidentiality, and non-repudiation. In order to fulfill such goals, we will discuss an experimental integration of the XMPP with both a) *Security Assertion Markup Language* and *Simple Authentication - Security Layer* (SAML/SASL) for server-to-server SSO authentication and b) XEP-0027 extensions for secure message exchange.

The paper is organized as follows. Section II describes the state of the art. Section III discusses the advantages of using XMPP-based communication systems for complex federation-enabled Cloud architectures. In Section IV, we highlight the limits of the XMPP in term of security. Possible security integration to the XMPP are discussed in Section V. More specifically, we will discuss how the Internet-Draft entitled "A SASL Mechanism for SAML", defined by CISCO TF-Mobility Vienna can be adopted to achieve secure federation between Cloud providers belonging to different administrative domains and how the XEP-0027 Specification can be adopted to achieve message signing/encryption for the inter-module communication. Section VI

concludes the paper.

II. Related Works

In this section, we describe the current state-of-the-art on Security and Cloud computing.

More works treating security and privacy on Cloud computing exist in literature. Many of them provide theoretical discussions on different security aspects, but a few try to find concrete assessments and real implemented solutions [5], [6], [7]. In [8], the authors identify security challenges to manage multi-provider Inter-Cloud environments with dedicated communication infrastructures, security mechanisms, processes and policies. The existing security challenges in Collaborative Clouds are highlighted in [9]. The authors analyze different security initiatives, such as the FCAPS model (ISO 10164), that is helpful for describing security management functionalities, and the ISO/IEC 27001, that offers a methodology for managing security services. We agree with the authors in [10], who assert that an Information Technology (IT) auditing mechanisms and framework can play an important role in compliance of Cloud IT security policies. In particular, a Third Party Auditor (TPA), introduced on behalf of Cloud users, has resources and experience that users do not have and it can be emplaced to audit the integrity of large data stored on Clouds. Their survey highlights the possibility to theoretically use recent technologies for enforcing security from different point of views, as well as SAML, OAuth, HMACs, homomorphic linear authenticators (HLAs), identity and access management as a service (IDaaS). As the architecture of IDS (Intrusion Detecting System), aimed at Nodes NIDS and Hosts HIDS. A collaboration-based Cloud computing security management framework is presented in [11]. The alignment of NIST-FISMA standard with the Cloud computing model is reported in a table form. The table reports the responsibilities of Service Providers (SPs), Cloud Customers (CCs) and Cloud Providers (CPs) in managing security assets. The work that authors described is interesting, because it offers a web portal along with a database where to track Cloud resources utilization and their security implications. Risks Management, where risk probabilities and vulnerability descriptions along with standards are reported. The authors in [12] describe a dynamic hierarchical role-based access control model, useful in Cloud service aimed at Mobile Internet. In particular, they introduce an interesting security model with self-adaptive features. Their self-adaptive schema enables their system to automatically meet the environment parameters, hence offering the corresponding protections. Another work that presents possible threats in Cloud is [13]. The authors highlight that when organizations migrate their data or services into the Cloud, they are not aware of their locations. Organizations lose control over their data and are not aware of any security mechanisms put in place by the Cloud provider. In this situation, the main three security concerns are: *Loss Of Control*, *Compliance Implications* and *Confidentiality and Auditing*. Our work focuses the attention on these issues.

III. Cloud Computing and XMPP

A valuable solution for the design of a performance and secure Cloud middleware is to conceive its communication system adopting an instant message-oriented approach. To this regard the XMPP (RFC 3920 and RFC 3921), also called Jabber, is becoming more and more popular due to its flexibility to suit different scenarios where a high level of re-activeness is strongly required. Despite it was born for human interaction via chat room it can be used to develop the communication of whatever distributed system well fitting the requirements of Cloud computing. XMPP is an XML-based protocol used for near-real-time, extensible instant messaging and presence information. XMPP remains the core protocol of the Jabber Instant Messaging and Presence technology. The "Jabber" technology leverages open standards to provide a highly scalable architecture that supports the aggregation of presence information across different devices, users and applications. Like email, anyone who has a domain name and an Internet connection can run the Jabber server and chat with other users. The Internet Engineering Task Force has formalized XMPP as an approved instant messaging and presence technology, and the specifications have been published as RFC 3920 and RFC 3921. Born for

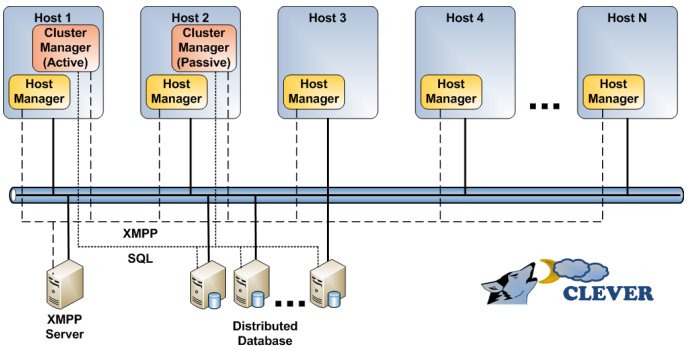

Figure 1. CLEVER architecture.

human interaction via chat the XMPP offers many advantages for the design of communication system of complex distributed system In the panorama of Cloud computing the XMPP represents a flexible solution because custom functionality can be built on top of XMPP, and common extensions are managed by the XMPP Software Foundation. The XMPP provides a technology for asynchronous end-to-end exchange of structured data. Considering a distributed system, the protocol allows to build one or more overlay networks having: global addressing (JIDs), network availability (presence), concurrent information transactions, distributed federated networks, structured data with XML payload.

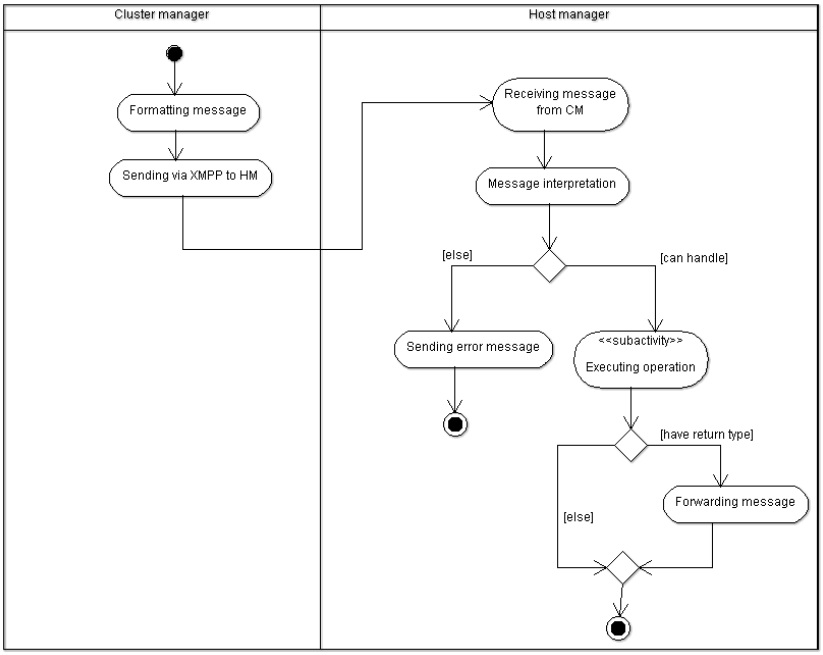

Figure 2. Activity diagram of the external communication.

The architecture is similar to the email network, but it introduces several added value features to facilitate near-real-time communications. The end-to-end communication in XMPP is logically peer-to-peer but logically client-to-server-to-server-to-client. If we assume that each server can manage a domain, a server-to-server connection can enable inter-domain federation. In the XMPP, data are sent over persistent XML steams. XMPP clients (i.e., human or software modules) are connected over a room that represents a sort of broadcast domain.

Summarizing, the XMPP presents several advantages compared to the HTTP-based web services including

- End-to-end communication
- It offers real-time capabilities such as heartbeat, alarms, and any kind of asynchronous communication.
- Efficient distribution of data with public/subscribe and direct-push approaches (e.g., configuration distribution, push RSS/Atom, data collection, log processing, fast results delivery software modules and clients.).
- Advance service discovery.
- Federation. Most firewalls allow users to fetch and post messages without hindrance. Thus, if the TCP port used by XMPP is blocked, a server can listen on the normal HTTP port and the traffic should pass without problems.

In order to describe how an XMPP-based communication system of a federation-enabled Cloud architecture works, in the following we will consider as model CLEVER [14], an Infrastructure as a Service (IaaS) middleware. The CLEVER middleware is based on the architecture schema depicted in Figure 1, which shows a cluster of n nodes (also an inter-connection 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 the entities interact exchanging information by mean of the Communication System based on the XMPP. The set of data necessary to enable the middleware functioning is stored within a specific Database deployed in a distributed fashion.

Figure 1 shows the main components of the CLEVER architecture, which can be split into two logical categories: the software agents (typical of the architecture itself) and the tools they exploit. To the former set belong both the Host Manager and the Cluster Manager:

- The Host manager (HM) performs the operations needed to monitor the physical resources and the instantiated VMs. It runs the VMs on the physical hosts and performs the migration of VMs.
- The Cluster Manager (CM) acts as an interface between the clients (software entities, which can exploit the Cloud) and the HM agents. It performs the management of VM images (uploading, discovering, etc.) and the monitoring of the overall state of the cluster (resource usage, VMs state, etc.).

Regarding the tools such middleware components exploit, we can identify the Distributed Database and the XMPP Server. The XMPP-based communication system allows to conceive more flexible inter-module communication and

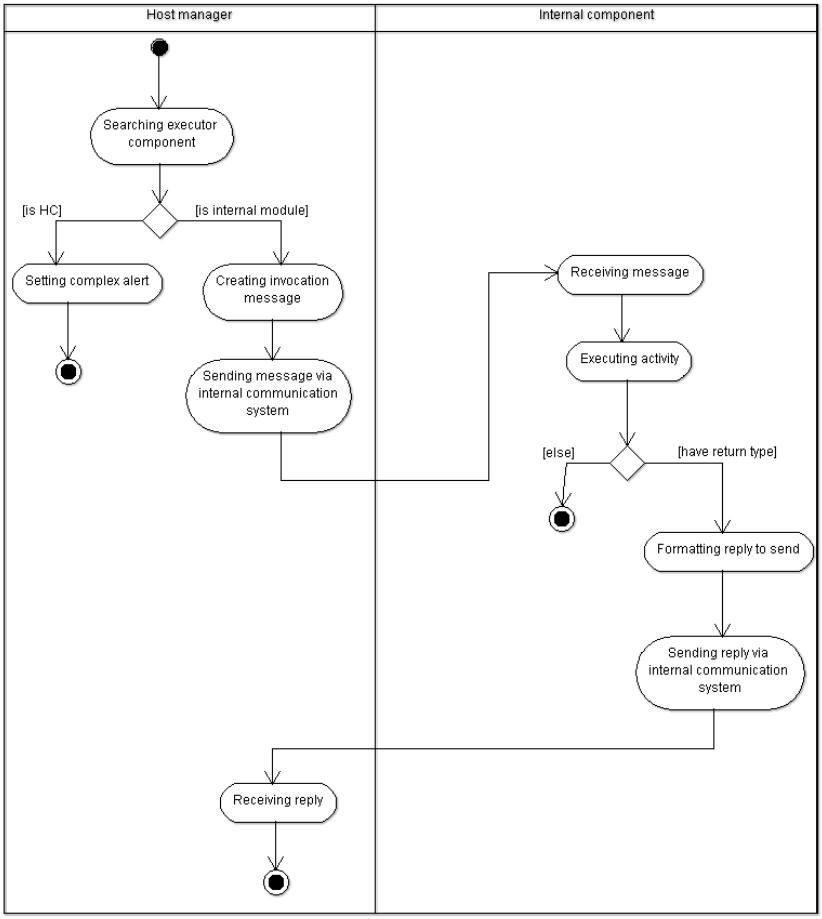

Figure 3. Activity Diagram of the sub-activity Executing Operation.

inter-domain federation capabilities compared to the HTTP-base web services.

A. *Inter-Module Communication*

When two different hosts have to interact each other, the inter-module communication has to be exploited. The typical use cases refer to:

- Communication between CM and HM for exchanging information on the cluster state and sending specific commands;
- Communication between the administrators and CM using the ad-hoc client interface.

As previously discussed, in order to implement the inter-module communication mechanism, an XMPP server must exist within the CLEVER domain and all its entities must be connected to the same XMPP room. When a message has to be transmitted from the CM to an HM, as represented in Figure 2, it is formatted and then sent using the XMPP. Once received, the message is checked from the HM, for verifying if the requested operation can be performed. As the figure shows, two different situations could lay before: if the request can be handled, it is performed sending eventually an answer to the CM (if a return value is expected), otherwise an error message will be sent specifying an error code. The "Execution Operation" is a sub-activity whose description is pointed out in Figure 3. When the sub-activity is performed, if any return value is expected the procedure terminates, else this value has to be forwarded to the CM in the same way has been done previously with the request. The sequence of steps involved in the sub-activity is represented in Figure 3. If the operation that has to be executed involves a component different from the Host Coordinator, the already described intra-module communication has to be employed. Once the selected component receives the message using this mechanism, if no problem occurs, the associated activity will be performed, else an error will be generated. If the operation is executed correctly and a return value has to be generated, the component will be responsible of generating the response message that will be forwarded to the HM, and thus, to the CM.

B. Inter-Domain Federation

CLEVER has been designed with an eye toward federation. In fact, the choice of using XMPP for the CLEVER module communication (i.e., external communication XMPP room) has been made thinking about the possibility to support in the future also interdomain communication between different CLEVER administrative domains. Federation allows Clouds to "lend" and "borrow" computing and storage resources to/from other Clouds. In the case of CLEVER, this means that a CM of an administrative domain is able to control one or more HMs belonging other administrative domains. For example, if a CLEVER domain A runs out of resources of its own HMs, it can establish a federation with a CLEVER domain B, in order to allow the CM of the domain A to use one or more HMs of the domain B. This enables the CM of domain A to allocate VMs both in its own HMs and in the rented HMs of domain B. In this way, on one hand the CLEVER Cloud of domain A can continue to allocate services for its clients (e.g., IT companies, organization, desktop end-users, etcetera), whereas on the other hand the CLEVER Cloud of domain A earns money from the CLEVER Cloud of domain B for the renting of its HMs.

As anyone may run its own XMPP server on its own domain, it is the interconnection among these servers that exploits the interdomain communication. Usually, every user on the XMPP network has an unique Jabber ID (JID). To avoid requiring a central server to maintain a list of IDs, the JID is structured similarly to an e-mail address with an 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@domainB.net, whereas a HM could be identified by a JID liszt@domainA.net: bach and liszt respectively represent the host names of the CM and the HM, instead domainB.net and domainA.net represent respectively the domains of the Cloud that "borrows" its HMs and of the Cloud that "lends" HMs. Let us suppose that bach@domainB.net wants to communicate with liszt@domainA.net, bach and liszt, each respectively, have accounts on domainB.net and domain A XMPP servers.

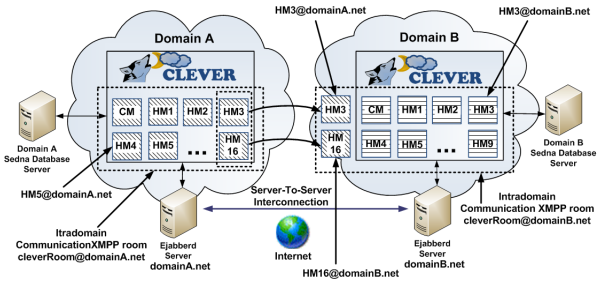

Figure 4. Example of federation between two CLEVER Clouds.

The idea of CLEVER federation is straightforward by means of the built-in XMPP features. Figure 4 depicts an example of interdomain communication between two CLEVER administrative domains for the renting of two HMs from a domain A to domain B. Considering the aforementioned domains, i.e., domainA.net and domainB.net, in scenarios without federation, they respectively include different XMPP rooms for intradomain communication (i.e., cleverRoom@domainA.net and cleverRoom@domainB.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. Considering a federation scenario between the two domains, if the CM the domainB.net domain needs of external resources, after a priori agreements, it can invite within its cleverRoom@domainB.net room one or more HMs of the domainA.net domain. For example, as depicted in Figure 4, the CLEVER Cloud of domainB.net rents from the CLEVER Cloud of domainA.net, HM6 and HM16. Thus, the two rented HMs will be physically placed in domainA.net, but they will be logically included in domainB.net. As previously stated, in order to accomplish such a task a trust relationship between the domainA.net and the domainB.net XMPP servers has to be established in order to enable a Server-to-Server communication allowing to HMs of domain A to join the external communication XMPP room of domain B.

IV. Security Issues in XMPP-based Clouds

According to the Domains and sub-topics investigated by the CSA, we worked on partial security needs for making a concrete solution aimed at IaaS level. The elements we identified in this assessment that are useful in the IaaS context are presented in the CSA guidance [15] and summarized as following:

1) In the Governance and Enterprise Risk Management, there is the need to *"divulge policies, procedures and processes comprising the Cloud providers' Information Security Management System (ISMS)"*, knowing who makes what.
2) Whereas in the Information Management and Data Security, it is necessary to *"assure that Cloud provider personnel controls are in place to provide a logical segregation of duties."*
3) In the Traditional Security, Business Continuity and Disaster Recovery, *"Customers should perform onsite inspections of Cloud provider facilities whenever possible."*
4) Data Center Operation, *"Cloud providers must all be able to demonstrate comprehensive compartmentalization of systems, networks, management, provisioning and personnel."*
5) In the Incident Response, Notification and Remediation, *"Cloud providers should construct a registry of*

application owners by application interface (URL, SOA service, etc.)".

6) Encryption and Key Management, where *"segregate the key management from the Cloud provider hosting the data, creating a chain of separation"*.

Even though the XMPP support the SASL and TLS technologies for the authentication and encryption of the communication channels between different eJabberd servers, it presents some security limitations due to the decentralized nature of the protocol that demands the accomplishment of specific security mechanisms to the different implementations. On the other hand, the flexible and extensible nature of the protocol allows to integrate basic security mechanisms, improving the level of the security in communication. In particular, considering federation-enabled Clouds, the XMPP does not allow to natively develop the following security mechanisms

- **Data Confidentiality, Integrity, and Non Repudiation for Inter-Module Communication**. As previously discussed, the different software modules can communicate over one or more chat-rooms. Chat-rooms allow to isolate the communication of the involved software modules also providing a way to control which module can join a chat-room by means of a username/password authentication. This level of security is particularly weak considering Cloud architectures. Considering software modules A and B of a Cloud system i) module A and B have to perform a mutual authentication before communicating through X.509 certificates in order to avoid identity-thief attacks; ii) message exchanged between two software modules have to be confidential and not corrupted in order to avoid man-in-the-middle attacks; iii) if software module A sends a message to B, module A cannot deny of having done so. In order to guarantee Data Confidentiality, Integrity, and Non Repudiation for Message Exchange further security mechanisms have to be integrated in the XMPP.
- **SSO Authentication for Inter-Domain Federation**. Federation between Cloud providers implies the establishment of a secure inter-domain communication between the XMPP servers of the involved Clouds. This raises several issues regarding the management of authentication between the XMPP servers of different Clouds. In fact, considering a scalable scenario including n Clouds in order to perform an inter-domain federation the XMPP server of each Cloud should perform $n-1$ authentication on the other $n-1$, hence managing $n-1$ different credentials for accessing the federated Clouds. Considering the whole Cloud federation ecosystem it is required to manage $n(n-1)$ different credentials. The IdP/SP scheme allows to address this problem introducing a trusted third-party, the IdP, so that a cloud that wants to perform a federation with the other $n-1$ Clouds has to perform the authentication once, gaining the access on the other $n-1$ Clouds, which will be trusted with the IdP. This form of federation is called SSO. Unfortunately, at the moment of the writing of this paper, the SASL/TLS on which the XMPP is based does not support any standard form of SSO Authentication for server-to-server federation.

In the following, we will deepen the previously introduced security limitations considering a federated Cloud computing scenario based on the XMPP.

A. XMPP Concerns Regarding Data Confidentiality, Integrity, and Non Repudiation for Inter-Module Communication

Let us consider a message oriented Cloud system including several distributed software modules or components and whose inter-module communication takes place by means of an instant messaging protocol. The question is: which are the security requirements of the involved communication system? Definitely it should ensure: *data confidentiality*, *data integrity*, and *data non-repudiation of the sender/receiver*. Let us assume that in order to achieve a totally secure communication system each message has to be signed and encrypted by each component. From now on, for simplicity,

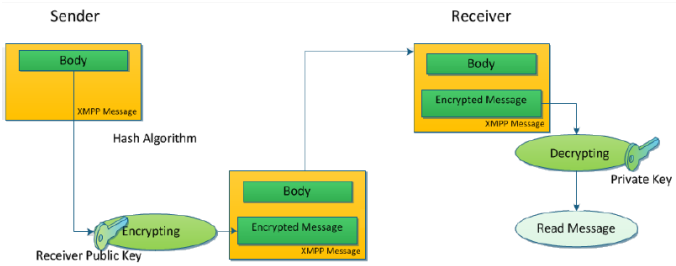

Figure 5. XMPP messages encryption in CLEVER.

we will focus our analysis on the XMPP as instant messaging protocol. Considering the aforementioned security requirements, XMPP as has to be properly extended. In our opinion, considering a Public Key Infrastructure (PKI), the XMMP-based communication of a message oriented Cloud system should support the following functionalities:

- **Digital identity management**. Each component during the in-band registration (i.e., an automatic enrollement of a client on the XMPP server) with the XMPP server requires a digital certificate to a trusted Certification Authority (CA) through the Simple Certificate Enrollment Protocol (SCEP).
- **Signed message exchange**. Each component should be able to sign a message sent to another one.

- **Encrypted message exchange**. Each component should be able to perform a total or partial encryption of a message.
- **Private chat rooms**. The communication system should allow the management of private chat room with restricted access to authorized components.
- **Encrypted chat rooms.** The communication system should allow the management of private and encrypted chat rooms. The key exchange between the communicating components should take place according to a PKI schema. The component that play the role of "moderator" instantiate a new chat room associating a session key. When a new component wants to join the communication, the "moderator" component sends the session key encrypted with the public key of the new component itself.

B. XMPP Concerns Regarding SSO Authentication for Inter-Domain Federation

Considering that the communication in each CLEVER Cloud is achieved through XMPP or Jabber messages by means of an Ejabberd server (i.e., an instant message server), the federation establishment between two or more CLEVER Clouds implies a secure server-to-server inter-domain communication between their respective Ejabberd servers. In fact, in the XMPP terminology, the term "federation" is commonly used to describe communication between two servers.

Thus, in CLEVER, each Cloud belongs to a domain managed by an XMPP server. In CLEVER, the way to federate two Clouds is to establish a server-to-server inter-domain communication between the XMPP servers to the involved Clouds.

Cloud federation raises many issues especially in the field of security and privacy. Single Sign On (SSO) authentication is fundamental for achieving security in a scalable scenario such as Cloud federation. However, the Simple Authentication and Security Layer (SASL) [16], i.e., a framework for authentication and data security in Internet protocols, supported by XMPP does not support any SSO authentication mechanism.

The public-subscribe technology is reemerging for enabling real-time communication within Cloud infrastructure, nevertheless its major protocol XMPP is somewhat dated from the point of view of security.

In order to enable federation between servers, it is needed to carry out a strong security to ensure both authentication and confidentiality thanks to encryption. According to the IETF 6120, compliant implementations of servers should support Dialback or SASL EXTERNAL protocol for authentication and the TLS protocol for encryption.

The basic idea behind Server Dialback [17] is that a receiving server does not accept XMPP traffic from a sending server until it has (i) "called back" the authoritative server for the domain asserted by the sending server and (ii) verified that the sending server is truly authorized to generate XMPP traffic for that domain. The basic flow of events in Server Dialback consists of the following four steps:

1) The Originating Server generates a Dialback key and sends that value over its XML stream with the Receiving Server. (If the Originating Server does not yet have an XML stream to the Receiving Server, it will first need to perform a DNS lookup on the Target Domain and thus discover the Receiving Server, open a TCP connection to the discovered IP address and port, and establish an XML stream with the Receiving Server.)
2) Instead of immediately accepting XML stanzas on the connection from the Originating Server, the Receiving Server sends the same Dialback key over its XML stream with the Authoritative Server for verification. (If the Receiving Server does not yet have an XML stream to the Authoritative Server, it will first need to perform a DNS lookup on the Sender Domain and thus discover the Authoritative Server, open a TCP connection to the discovered IP address and port, and establish an XML stream with the Authoritative Server).
3) The Authoritative Server informs the Receiving Server whether the key is valid or invalid.
4) The Receiving Server informs the Originating Server whether its identity has been verified or not.

SASL is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. It provides a structured interface between protocols and mechanisms. The resulting framework allows new protocols to reuse existing mechanisms and allows old protocols to make use of new mechanisms. SASL is used in various application protocols (e.g., XMPP, IMAP, LDAP, SMTP, POP, etc.) and support many mechanisms including:

- **PLAIN**, a simple clear text password mechanism. PLAIN obsoleted the LOGIN mechanism.
- **SKEY**, an S/KEY mechanism.
- **CRAM-MD5**, a simple challenge-response scheme based on HMAC-MD5.
- **DIGEST-MD5**, HTTP Digest compatible challenge-response scheme based upon MD5. DIGEST-MD5 offers a data security layer.
- **GSSAPI**, for Kerberos V5 authentication via the GSS-API. GSSAPI offers a data-security layer.
- **GateKeeper**, a challenge-response mechanism developed by Microsoft for MSN Chat

At the time of writing of the IETF 6120, in March 2011, most server implementations still use the Dialback protocol to provide weak identity verification instead of using SASL to provide strong authentication, especially in cases where SASL negotiation would not result in strong authentication anyway (e.g., because TLS negotiation was not mandated

by the peer server, or because the PKIX certificate presented by the peer server during TLS negotiation is self-signed and has not been previously accepted). The solutions is to offer a significantly stronger level of security through SASL and TLS.

V. Securing XMPP-Based Communication System for Federation-Enabled Clouds

A. A Solution for Secure Message Exchange in Inter-Module Communication

Custom functionality can be built on top of XMPP, and common extensions are managed by the XMPP Software Foundation. Regarding the security, even though the XMPP specification support the SASL and TLS technologies for the authentication and encryption of the communication channels, it presents some limitations due to the decentralized nature of the protocol that demands the accomplishment of specific security mechanisms to the different implementations. On the other hand, the flexible and extensible nature of the protocol allows to integrate basic security mechanisms, improving the level of the security in the communications.

As previously discussed, in order to guarantee data confidentiality, integrity, and non repudiation in the XMPP-based communication system of a federation-enabled Cloud Architecture specific security extensions are required. The *XEP 0027* [18] specification describes the use of Jabber with the Open Pretty Good Privacy (OpenPGP - RFC 4880 - [19]). OpenPGP is an interoperable specification that provides cryptographic privacy and authentication for data communications. As highlighted by the *internet draft*, the XEP 0027 does not represent a standard, although it could be in the future, but it describes a possible solution for authentication and data encryption in end-to-end XMPP communication. XEP 0027 allows the addition of specific XML tags in the XMPP message, each one defined by a specific *namespace*: for example *"jabber:x:signed"* and *"jabber:x:encrypted"*. Such tags, indicate to the system how to process the information contained within them. As suggested in the specification, it is possible to apply the digital sign of a sender to the message, for example calculating a *message digest* and signing it with his/her private key. The specification, also allows to sign the presence message in a chat room. In this case, it is possible to sign the status of the sender. In the following, it is shown an example of XMPP message sent from the *Alice* to *Bob* user.

```
<presence from='alice@example.com'
        to='bob@example2.com'>
  <status>Online</status>
  <x xmlns='jabber:x:signed'>
     iQA/AwUBOjU5dnol3d88qZ77EQI2JAC
     fRngLJ045brNnaCX78ykKNUZaTIoAoP
     HI2uJxPMGR73EBIvEpcv0LRSy+=45f8
  </x>
</presence>
```

The status of *alice* is signed with her private key, so that *bob* can verify by means of the *Alice*'s public key that she is really online. In the same way, it is possible to encrypt the content of the tag body using the public key of the receiver in order to achieve confidentiality. In the following, it is shown a message sent from *Alice* to *Bob* whose content has been encrypted with the public key of *Bob*.

```
<message to='alice@example1.com'
        from='bob@example2.com'>
  <body>This message is encrypted.</body>
    <x xmlns='jabber:x:encrypted'>
      qANQR1DBwU4DX7jmYZnncmUQB/9KuKBdd
      zQH+tZ1ZywKK0yHKnq57kWq+RFtQdCJWp
      dWpR0uQsuJe7+vh3NWn59/gTc5MDlX8dS
      9p0ovStmNcyLhxVgmqS8ZKhsblVeuIpQ0
      JgavABqibJolc3BKrVtVV1igKiX/N7Pi8
      RtY1K18toaMDhdEfhBRzO/XB0+P
    </x>
  </body>
</message>
```

The specification does not define the exchange of public keys that is demanded to OpenPGP. Even though the chat messaging is something that purely seem regarding the human interaction, the same approach can be applied to Cloud computing systems in which different distributed software components need to interact each others in both real time and in secure way.

In order to secure the inter-module communication, it is needed to integrate PKI, SCEP, CA, and LDAP mechanism. In order to achieve a secure inter-module communication, it is mandatory to have a digital identity for each element. For this reason, during the initialization of each entity (e.g., administration client, CM, or HM module), it is needed to setup the corresponding digital identity. Each entity obtains through the SCEP a private/public key pair from the CA. After that, it creates a KeyStore local object, in which each requesting entity can find, protected by password, its private key and the digital certificate in PKCS# format. After that the certificate is published on the LDAP server acting as "publisher" of the digital certificates associated to the various entities.

When a module has obtained its own digital identity and it can access the LDAP server storing the public keys of the other entities, it is able to establish a secure inter-module communication with other modules. Thus, each module will be able to sign a message with its private key and to encrypt target message contents. In the first case, the receiver module will verify the digital sign of the sender by means of the corresponding public key read from the LDAP server. In the second case, a module will be able to use the PKI infrastructure in order to negotiate a shared key in order to encrypt the date according to a symmetric cryptography scheme (we remark that the symmetric encryption is more performing than an asymmetric one from a computational point of view). Figure 6 and Figure 7 show the activity diagrams of the inter-module communication respectively

in plain text and with authentication/encryption. The basic difference from the two activity diagrams consists in an encoding task before the message sending and in a decoding task after the reception of the message.

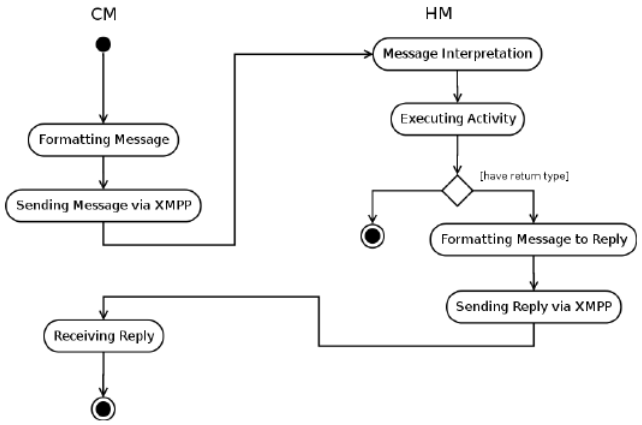

Figure 6. Inter-module communication without security.

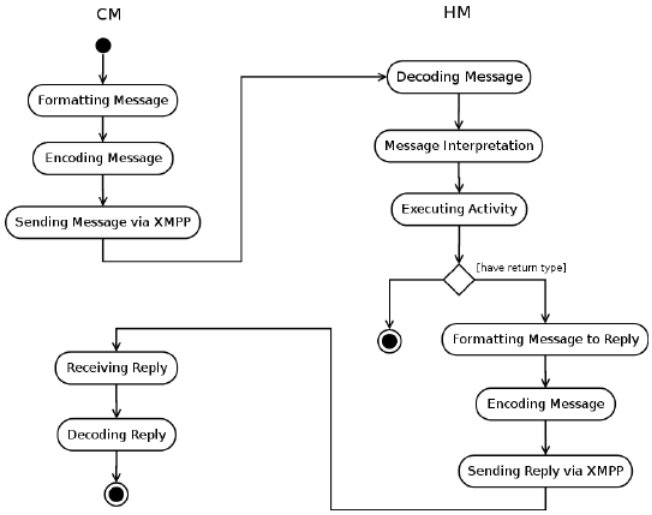

Figure 7. Inter-module communication with security.

In order to guarantee data confidentiality, integrity, and non repudiation in the XMPP-based communication system of a federation-enabled Cloud Architecture, four basic extensions are required in the XMPP messages:

- **Signed**. It allows to attach to the message body a digest signed with the private key of the sender component. The signed extension is identified by the XML name space jabber:x:signed (*<x xmlns='jabber:x:signed'>*) known by all the components. When the message arrives to the receiver component, it detects the signed extension and it queries the LDAP publisher server if an X509 certificate exists for the sender. If it exists, the receiver validates the sign and verifies the message digest according to a shared algorithm.
- **Encrypted**. It allows to attach to the message body a content encrypted with the public key of the receiver component. When a component wants to send an encrypted message, it requests to the LDAP publisher server the X509 certificate of the receiver component. Thus using the public key of the receiver, the sender encrypts the message and it includes the encryption extension identified by the "jabber:x:encrypted" name space (*<x xmlns='jabber:x:encrypted'>*). When the message arrives to destination, the receiver component decrypts it with its private key. This process is schematized in Figure 5.
- **Session Key**. It allows to attach the message a session key. It is used to support an hybrid encryption scheme: the unique share key or the session key is used to encrypt/decrypt the messages by both sender and receiver according to a symmetric encryption scheme (already used in the SSL/TLS protocol), but the session key is exchanged between the two parties according to a public key or asymmetric schema. The advantages of such a hybrid cryptographic scheme is twofold: session key secrecy and faster processing during the encryption of the message.
- **Timestamp**. It allows to attach to the message a signed timestamp, in order to make the message oriented Cloud system normative compliant.

B. A Solution for SSO Authentication in Inter-Domain Federation

In a scalable scenario of federation, each Cloud can require to frequently establish/break partnerships with other Clouds. This implies that each Cloud should manage a huge number of credentials in order to authenticate itself in other Clouds. In a federated environment, this means that the XMPP server of the Cloud requiring federation has to be authenticated by the XMPP server of the Cloud accepting the federation request. If we consider thousand of Clouds, each Cloud should manage one credential for accessing to each federated Cloud. This is problem is commonly known as Single-Sign-One (SSO), i.e., considering an inter-domain environment, performing the authentication once, gaining the access to the resources supplied by different Service Provider, each one belonging to a specific domain. A model addressing the SSO problem is the Identity Provider/Service Provider Model (IdP/SP). Typically, a client who wants to access to the resources provided by a SP performs the authentication once on the IdP (asserting party), which asserts to the SP (relaying party) the validity of the authentication of the client. Considering many SPs relaying on the IdP if the client wants to access another SP, as this latter will be trusted with the IdP, no further authentication will be required. This

model is widely known on the Web with the term "Web Browser SSO", in which the client is commonly an user who perform an authentication fill in an HTML form with his user name and password. Nowadays, the major standard implementing defining the IdP/SP model is the Security Assertion Markup Language (SAML) [20], developed by OASIS.

The scenario of federation is quite similar. In this case, the client who wants to perform the authentication is the XMPP server of the Cloud requiring federation, instead the role of the SP is played by the XMPP server of the Cloud accepting the federation request. As the XMPP server support authentication through SASL a concern raises: the RFC 4422 does not support any security mechanism implementing the IdP/SP model. Therefore, in order to achieve such a scenario, we followed the Internet-Draft entitled "A SASL Mechanism for SAML", defined by CISCO TF-Mobility Vienna, describing the applicability and integration between the two protocols for non-HTTP use cases. According to such a draft, the authentication should occur as follows:

1) The server MAY advertise the SAML20 capability.
2) The client initiates a SASL authentication with SAML20
3) The server sends the client one of two responses:
   a) a redirect to an IdP discovery service; or
   b) a redirect to the IdP with a complete authentication request.
4) In either case, the client MUST send an empty response.
5) The SASL client hands the redirect to either a browser or an appropriate handler (either external or internal to the client),and the SAML authentication proceeds externally and opaquely from the SASL process.
6) The SASL Server indicates success or failure, along with an optional list of attributes

In this way, thanks to SASL and SAML, for each Cloud it is possible to perform the authentication once gaining the access to all the other Clouds relying on the IdP, thence, lending and/or borrowing HMs according to agreements.

In a Cloud Federated scenario, in which the communication system of each involved Cloud is based on the XMPP, the most convenient and easy way for the establishment of a federation is the employment of the federation features made available by the XMPP protocol itself. This latter assumes a XMPP server can be configured for accepting external connections from other servers for creating server-to-server interactions (server federation).

According to the XMPP specifications, this mechanism is quite easy to implement and the result will be the ability for two XMPP servers in different domains to exchange XML stanzas. There are different levels of federation:

- Permissive Federation, a server accepts a connection from any other peer on the network, even without verifying the identity of the peer based on DNS lookups.
- Verified Federation, a server accepts a connection from a peer only after the identity of the peer has been weakly verified via Server Dialback, based on information obtained via the Domain Name System (DNS) and verification keys exchanged in-band over XMPP.
- Encrypted Federation, a server accepts a connection from a peer only if the peer supports Transport Layer Security (TLS) and the client authenticates itself using a SASL mechanisms.

On one hand, Permissive and Verified Federation are the simplest federation approaches: as discussed in the previous section, they lack some security aspects since they are not based on any password exchange procedure and, in order to implement domain filtering (in the second case), a list of allowed sites has to be compiled preemptively. On the other hand, the Encrypted Federation level relies on a more secure way to perform the authentication, based on challenge-response authentication protocols relaying on passphrase.

This standard authentication mechanisms are enough when you want to enable the communication among a limited endpoint number but, in a scenario where several XMPP servers might exist, it could be a difficult task to statically pre-configure the binding among all the involved entities and manage credentials for authenticating a given server to each other. Our idea aims to address these issues and propose the integration of a new SASL security mechanism for allowing a more scalable management of the authentication process exploiting the well-known concept of SSO. The integration we are talking about refers to the use of SAML 2.0.

In order to discuss how to achieve the above mentioned scenario, we two hypothetical Cloud providers each one relying on its own Ejabberd XMPP server [21] to allow communication within its domain. Generally, the server-to-server federation is accomplished by an Ejabberd module that manages incoming and out-coming connections from/to external servers. According to the XMPP core specification, this module is able to establish server federation according to the three different levels pointed out above. In order to enable Ejabberd servers to perform SSO authentication it is required to consider the Encrypted Federation case and extending the Ejabberd module performing SASL to add in the list of the supported security mechanism also SAML 2.0.

A possible way for the achievement of this environment is the implementation of the Internet-Draft entitled "A SASL Mechanism for SAML" defined by CISCO TF-Mobility Vienna relying on an external software module based on Shibboleth, we named Authentication Agent (AA). The AA acts as user when it is contacted from the Source Ejabberd Server for starting the Federation, whereas represents the Relying Party when it is contacted from the Destination Ejabberd Server.

In the following, we present the sequence of steps performed by two servers (for simplicity Source Server and

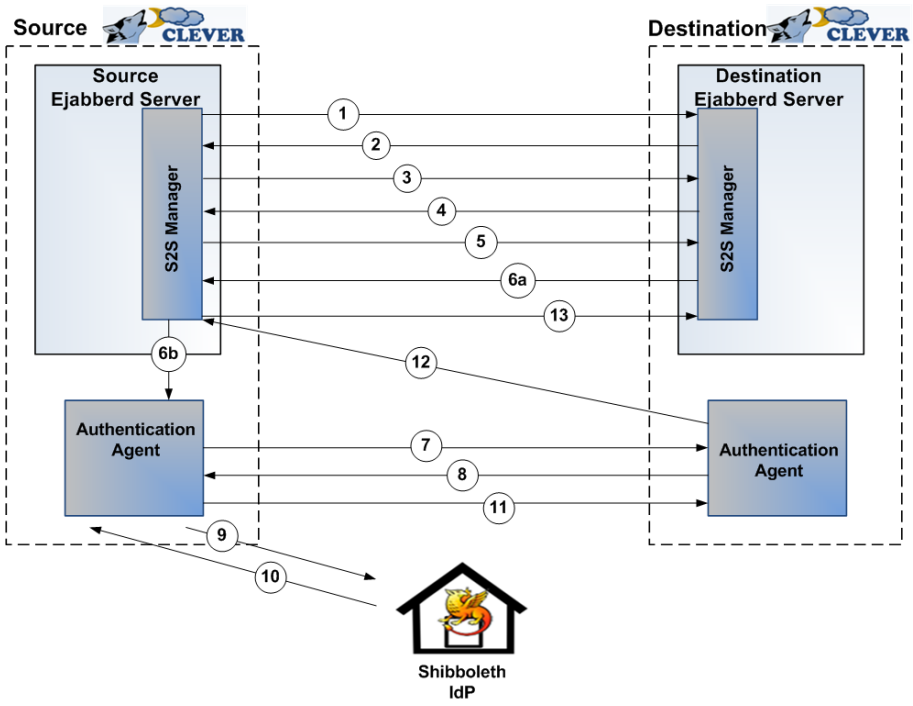

Figure 8. Step performed by two XMPP servers aiming to build Federation: the authentication process is executed using SAML 2.0 as external SASL mechanism.

Destination Server) that aim to build the federation. As Figure 8 depicts, the involved actors in the process are the s2s_Manager(s) of both the Ejabberd servers, the two authentication agents acting as User and Relying Party (User, the one interacting with the Source Server; Relying Party the one interacting with the Destination Server) and the Identity Provider (also implemented using Shibboleth).

- Step 1: s2s_Manager of Source Server initiates stream to the s2s_Manager of the Destination server.
- Step 2: s2s_Manager of the Destination Server responds with a stream tag sent to the s2s_Manager of the Source Server.
- Step 3: s2s_Manager of the Destination Server informs the s2s_Manager of the Source Server of available authentication mechanisms.
- Step 4: s2s_Manager of the Source Server selects SAML as an authentication mechanism.
- Step 5: s2s_Manager of Destination Server sends a BASE64 encoded challenge to the s2s_Manager of the Source Server in the form of an HTTP Redirect to the Destination AA (acting as Relying Party).
- Step 6: a) s2s_Manager of Source Server sends a BASE64 encoded empty response to the challenge and b) forward to the Source AA the URL of the Relying Party.
- Step 7: The Source AA (User) engages the SAML authentication flow (external to SASL) contacting the Destination AA (Relying Party).
- Step 8: Destination AA redirect Source AA to the IdP.
- Step 9: Source AA contacts IdP and performs Authentication
- Step 10: IdP responds with Authentication Assertion
- Step 11: Source AA contacts Destination AA for gaining access to the resource.
- Step 12: Destination AA contacts the s2s_Manager of the Destination Server informing it about the authentication result.
- Step 13: if the authentication is successful the s2s_Manager of the Source Server initiates a new stream to the s2s_Manager of Destination Server.

The advantage of performing the authentication among servers in such a way mainly consists in the higher security level achieved than the traditional Dialback/SASL mechanisms and in the possibility of exploiting the SSO authentication. Looking at Figure 8, after that the federation has been achieved with the depicted server, if the same Source Cloud aims to perform server-to-server federation with a new XMPP server that relies on the same IdP as trusted third-party, such a process would be straightforward. Since the Source Server already has an established security context with the IdP, once the SASL process starts and the SAML mechanism is selected, no further authentication will be required.

VI. Conclusion

Currently, the major Cloud solutions base their communication systems on HTTP-based web services that do not well suit the requirements of the new emerging Cloud architectures and services. The XMPP presents several advantages for designing the communication system of a federation-enabled Cloud architecture. On one hand, the XMPP well suits the requirement of interactivity, but on the other hand it lacks of native security features. Both the inter-module communication and the inter-domain federation require appropriate security mechanisms. In this paper, we discussed two possible solutions to achieve such goals. Regarding the inter-module communication, we discussed how to extend the XMPP for enabling secure message exchange according to the XEP-0027 specification, whereas, considering the inter-domain federation, we proposed an approach based on SSO authentication for server to server federation according to the Internet-Draft entitled "A SASL Mechanism for SAML", defined by CISCO TF-Mobility Vienna. We hope, we contributed to alleviate the gap in security for the adoption of the XMPP in federation-enabled Cloud architectures.

References

[1] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "Federation between clever clouds through sasl/shibboleth authentication," in *INTERNET 2012, The Fourth International Conference on Evolving Internet*, pp. 77–84, 2012.

[2] A. Celesti, M. Fazio, and M. Villari, "Se clever: A secure message oriented middleware for cloud federation," in *IEEE Symposium on Computers and Communications (ISCC)*, July 2013.

[3] National Institute of Science and Technology. Standards Acceleration to Jumpstart Adoption of Cloud Computing; http://csrc.nist.gov/groups/SNS/cloud-computing/ July 2011.

[4] P. Saint-Andre, "Xmpp: lessons learned from ten years of xml messaging," *Communications Magazine, IEEE*, vol. 47, no. 4, pp. 92–96, 2009.

[5] W. Liu, "Research on cloud computing security problem and strategy," in *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pp. 1216 –1219, April 2012.

[6] A. Behl and K. Behl, "Security paradigms for cloud computing," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2012 Fourth International Conference on*, pp. 200–205, July 2013.

[7] A. Celesti, N. Peditto, F. Verboso, M. Villari, and A. Puliafito, "Draco paas: A distributed resilient adaptable cloud oriented platform," in *IEEE 27th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, pp. 1490–1497, 2013.

[8] M. Kretzschmar, M. Golling, and S. Hanigk, "Security management areas in the inter-cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 762 –763, July 2011.

[9] M. Kretzschmar and S. Hanigk, "Security management interoperability challenges for collaborative clouds," in *Systems and Virtualization Management (SVM), 2010 4th International DMTF Academic Alliance Workshop on*, pp. 43 –49, oct. 2010.

[10] I. Gul, A. ur Rehman, and M. Islam, "Cloud computing security auditing," in *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on*, pp. 143–148, June.

[11] M. Almorsy, J. Grundy, and A. Ibrahim, "Collaboration-based cloud computing security management framework," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 364 –371, July 2011.

[12] Z. Lian-chi and X. Chun-di, "Cloud security service providing schemes based on mobile internet framework," in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, vol. 3, pp. 307 –311, March 2012.

[13] A. Behl, "Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation," in *Information and Communication Technologies (WICT), 2011 World Congress on*, pp. 217–222, December 2011.

[14] A. Celesti, F. Tusa, M. Villari, and A. Puliafito., "Integration of CLEVER Clouds with Third Party Software Systems Through a REST Web Service Interface," in *17th IEEE Symposium on Computers and Communication (ISCC'12)*, 1-4 July 2012.

[15] Cloud Security Alliance, Security Guidance for Critical Areas of Focus in Cloud Computing v3.0, https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf, 2011.

[16] RFC 4422, Simple Authentication and Security Layer (SASL), http://www.ietf.org/rfc/rfc4422, Jun 2013.

[17] XEP-0220: Server Dialback, http://xmpp.org/extensions/xep-0220.html, Jun 2013.

[18] XEP-0027: Current Jabber OpenPGP Usage, http://xmpp.org/extensions/xep-0027.html, 2006.

[19] OpenPGP Message Format, http://www.rfc-editor.org/info/rfc4880, 2007.

[20] SAML V2.0 Technical Overview, OASIS, http://www.oasis-open.org/specs/index.php#saml, Jun 2013.

[21] Ejabberd, the Erlang Jabber/XMPP daemon: http://www.ejabberd.im/, Jun 2013.