Identity Management in Cloud Platforms using VOMS and SPID

Francesco De Angelis, Fausto Marcantoni, Alberto Polzonetti and Samuele Rilli
Computer Science Division, University of Camerino, Palazzo Battibocca, Via del Bastione, 62032, Camerino (MC), Italy

Keywords: Cloud Computing, Identity Management, VOMS, SPID, Cloud Foundry, Openstack, SAML, Authentication, Authorization.

Abstract: Cloud computing is being adopted more and more in recent years. It offers several benefits, such as high elasticity, availability and cost reduction, but yet presents some issues. Among the most important, the potential lack of security can affect the spreading of this technology. As cloud computing is pushing forward to the digital era, where users can have their own digital identity to access restricted resources or services, a reliable authentication and authorization system would attract more users to get involved in such process. This paper proposes an integration of the VOMS (Virtual Organization Membership Service) system for authorization and SPID (Sistema Pubblico per la gestione dell'Identità Digitale) system for authentication, within Cloud Foundry PaaS (Platform as a Service) model. Considerations, differences and interoperability matters will be addressed in order to provide a comprehensive scheme.

1 INTRODUCTION

Cloud computing, or simply cloud, is a word that is getting more and more used in the last few years. Although it is a novel technology, it is changing the way we are used to thinking about programs and computing. Cloud computing, in fact, aims to "shift IT services away from local computers to the Internet or, generally speaking, in networks. The network will be the computer" (T-Systems Enterprise, 2010). This means, many next generation applications and resources will be available directly on the Internet using a browser, rather than on the local machine. Nevertheless, issues affecting cloud computing are still slowing down this revolution: among the most important, we can point out the lack of standards, that affects the interoperability among different providers, as well as the security mechanisms for managing digital identities and the authentication process.

In this paper we study the integration of Cloud Foundry, a PaaS system, and OpenStack, an IaaS system, with authorization and authentication frameworks named VOMS (Virtual Organization Membership Service) and SPID (Sistema Pubblico per la gestione dell'Identità Digitale) respectively. The former is a tool used mainly in grid computing since several years. The latter is a system, being developed by the Italian government, for federated strong authentication; it is scheduled to enter service in April 2015, and complies with eIDAS European regulation, thus provides a framework interoperable not only in the Italian territory.

The remainder of the paper is organized as follow. Section 2 defines the IaaS and PaaS systems, including their main features. The concepts of identity and the relative identity management are introduced in Section 3, in particular the definition of authentication and authorization, and the presentation of VOMS and SPID models. In Section 4 we consider the implementation of a model that incorporates VOMS as authorization and SPID as authentication systems.

2 IAAS AND PAAS TOOL FOR CLOUD

A more technical definition of cloud computing is given by (Armbrust et al., 2010) as "both the applications delivered as services over the Internet, and the hardware and systems software in the data centers that provide those services".

Indeed, cloud providers supply IT resources in terms of infrastructure, software, storage and bandwidths, somehow like water and electricity are daily supplied in our houses. Computing capability is offered on demand to those companies that need flexible IT resources, and then they pay only for the time resources have been used, potentially leading to cost savings.

De Angelis F., Marcantoni F., Polzonetti A. and Rilli S..
Identity Management in Cloud Platforms using VOMS and SPID.
DOI: 10.5220/0005450800960103
In *Proceedings of the 11th International Conference on Web Information Systems and Technologies* (WEBIST-2015), pages 96-103
ISBN: 978-989-758-106-9

2.1 Cloud Models

Since cloud is a complex technology, it has been layered by NIST (NIST, 2011) in three main models.

SaaS (Software as a Service): supplies the user with the provider's application, and users do not manage or control the underlying cloud infrastructure including network, operating system, and application capabilities. An example of SaaS application is the e-mail service such as Google mail, where the customer simply uses the application, but doesn't know the software structure and is not responsible for its maintenance.

PaaS (Platform as a Service): the provider supports users with a framework for application development, so that they can build and deploy their own software. Users do not manage or control the underlying cloud infrastructure including network, operating system, and application capabilities, but have control over the deployed applications and possibly environment configurations.

IaaS (Infrastructure as a Service): user is provided with fundamental computing resources that he can use to deploy and run arbitrary software. Users don't manage or control the underlying cloud infrastructure but have control over operating system, storage, deployed applications, and possibly limited control over some networking components.

Several other "as a Service" are emerging recently, most notable are Database as a Service (DBaaS) and Backup as a Service (BaaS). The trend is to shift any functionality to a service provided through the Internet, that will bring to the dawn of the XaaS, "Anything as a Service" (Dixon, 2014).

Some examples of PaaS software are OpenShift (OpenShift, 2014), Salesforce (Salesforce, 2014), AppScale (AppScale, 2014), CloudControl (CloudControl, 2014), Cloud Foundry (Cloud Foundry, 2014), Microsoft Azure Web Sites (Azure, 2014). Regarding the IaaS, instances of well known systems are Amazon Elastic Compute Cloud (Amazon EC2, 2014), OpenStack (OpenStack, 2014), Apache Hadoop (Apache Hadoop, 2014), OpenNebula (OpenNebula, 2014).

Among all these options, we chose to use Cloud Foundry as PaaS and OpenStack as IaaS, because they are currently the leading projects in their category. Both are open source, developed by several different contributors, and supported by a broad community; more information are provided in Section 2.3.

2.2 Benefits and Downsides

Such technology can bring several benefits to users. At first, a cost saving factor for a company. A user can rent IT resources from a cloud instead of buying a physical machine, allowing to start using few resources at first and increase only when there is a further need. In this way the company avoids the initial hardware costs.

Then, the pay-as-you-go offer lets consumers pay only for the used computing resources on a short-term basis (for example, processors or storage resources by hours), and release them when are not needed anymore. The result is an optimization of resource usage, instead of having an own server idle during low workload periods, thus preventing the overprovisioning issue. Another aspect is the flexibility provided by the virtually infinite computing capacity that cloud makes available; this relieves the customer from the task to foresee near future resource needs and relative resource provision plans, avoiding the underprovisioning issue. Finally, the cloud data center itself manages the underlying infrastructure and technical problems, so that customers do not have to concern about IT maintenance nor to acquire a wide hardware competence.

However, significant challenges are yet to be addressed. Cloud is a young computing model and many systems are still in a development state: a lack of standard Application Public Interfaces (API) brings providers to use proprietary interfaces in their services, thus restricting the ability for consumers to move from a provider to another.

2.3 Cloud Foundry and OpenStack

Cloud Foundry is an open source cloud computing Platform as a Service (PaaS) being developed by several contributors, among which EMC, IBM, Rackspace, and VMware can be mentioned. Cloud Foundry is designed to support application development with high productivity, taking into account SaaS integration: it brings innovation in application services and at the same times incorporates heterogeneous cloud deployment options to facilitate migration. As overviewed by (Heller, 2014), Cloud Foundry provides easy to install frameworks that support languages such as Java, Node.js, Ruby and Go. Once an application has been deployed, Cloud Foundry stores it in an image which then is run within a container.

OpenStack is an open source system that provides Infrastructure as a Service (IaaS) functionalities. It comprises a series of components such as: Keystone for identity management, Nova for computing, Neutron for networking, Glace for image

storing, Horizon for dashboard, Swift for storage, Ceilometers for telemetry and billing accountability, Heat for orchestration. As an IaaS system, OpenStack can be installed over the bare hardware; it allows to create, launch, monitor and terminate instances of virtual machines, connect them over virtual networks, and mount virtual storage drives. Cloud Foundry is one of those instances that can be deployed over OpenStack.

3 IDENTITY MANAGEMENT

Another arguable aspect is the security of cloud. Many of the applications available on the internet usually require an authentication from the user. How a service manages and stores the customers identity is a very concerning issue. Privacy and data protection are matters that cannot be taken lightly, as users may need to provide their own personal information to access a service, and the system must assure the confidentiality of such data.

3.1 Authentication Vs Authorization

In a scenario where a user needs to authenticate, the idea is to set up a centralized system, where digital identities are managed by an external entity called identity provider (IdP), instead of entrusting the resource provider (the entity which supplies resources) itself with this functionality. The users identities are stored in the IdP, and a user can authenticate against the IdP to obtain his own digital identity. Additionally, including the IdP in a federated environment, is it possible to extend the centralized authentication to each cloud in the federation. For the users, this means they can own a single digital identity, and they can authenticate with that digital identity in any cloud belonging to the federation. Regarding resource providers, they won't have to store, secure and assure the privacy of the users information, as this work is delegated to IdPs.

In such context, we can point out the concepts of authentication and authorization. The former deals with the user identity: it permits to verify a person as he login to an application. The latter allows to grant permissions to carry out a given action, and applies both to users and to processes that must access a protected resource (Alfieri et al., 2005).

Although these two operations may feel very similar, actually they are pretty different. A user authenticates against a website to access to his account. Once logged, the user gains full control and thus can perform any action enabled for that account. In contrast, the user can authorize a third party application or website to use some functionalities or to act as behalf of the user. In this case, the third party application is not provided with user credentials, but instead with a token it can use to request access to resources or functionalities. The above approach is a step forward to improve security, and it allows a user to have a single digital identity instead of creating a second account in the third party website.

This means more control for the user, as he can select and restrict the allowed operations for the third party app, and at the same time the third party collects only user's token instead of his credentials. If the third party gets hacked, the hacker will be able to perform only the operations permitted by the user. to the third party. To solve this, the user can login to the main site and revoke access granted to the third party application, and no other action on the original website, such as password reset, is needed.

Authentication process can be performed using different kind of information: a password (something the user knows); smart cards that contain the user identifier (something the user has); a physical trait or characteristic that acts as user identifier, such as fingerprints and voice recognition (something the user is). Currently the most used techniques are the former (usually with username and password) and the second (typically a certificate contained in a smart card, or a OTP token).

In recent years the use of double factor authentication is increasing, since it offers a stronger security: smart cards are more difficult to be cloned than a password to be stolen. However this solution requires the user to always bring with him the smart card containing the identifier token. On the other hand, the password-base authentication remains the wider adopted solution, and reason is the simplicity of the process: users only need to remember their credentials, without the need of having any additional card that provides a certificate or device to receive an OTP. Furthermore, this is the most technologically neutral solution, as it do not require any specific device to read user's smart card.

Some examples of authentication systems used in cloud are: OpenID (OpenID 2015), User Account and Authentication (UAA, 2015), DIGIPASS as a Service (DIGIPASS, 2015), PowerBroker Open (PowerBroker Open, 2015). Instances of authorization tools are Conjur (Conjur, 2015), UAA, and OAuth 2.0 (OAuth 2.0, 2012).

3.2 VOMS

VOMS (Virtual Organization Membership Service) (VOMS, 2014) is defined by (Venturi et al., 2008) as

a tool that allows to define a dynamic collection of individuals, institutions, and resources, in order to flexible, secure, coordinated resource sharing across dynamic, multi-institutional collaborations. It is based on the concept of Virtual Organization (VO) that defines virtual collections, and adds functionalities to manage these abstract entities. It permits a resource owner to define a set of rules for sharing his resources, which can be used to drive authorization decisions. Such characteristic fits well in the grid computing system, an indeed VOMS is currently the de-facto standard for VO management: it is already being used by Lightweight Middleware for Grid Computing (gLite) now developed by EMI (EMI, 2014), and the Virtual Data ToolKit (VDT) (VDT, 2014) grid infrastructures.

3.2.1 VOMS Architecture

In order to provide its functionalities, the VOMS supplies several interfaces with different features (Alfieri et al., 2004):

- User client: sends a request to the User server containing user credentials (typically a X.509 certificate) and obtains a list of groups, roles and capabilities of the user;
- User server (we will refer to it more generally as VOMS server): receives requests from a user client and returns information about the user signed with the server X.509 certificate;
- Administration Client: is the tool used by VO administrators to manage users, groups, roles, etc;
- Administration Server: receives the requests from administration clients and accordingly updates the data in the system.

The server is essentially a front-end to an RDBMS, where all the information about users is stored. It acts as an Attribute Authority; the user sends to the VOMS server a request for attributes, and authenticates using his certificate. The VOMS server creates signed assertions containing the user's requested VO attributes, according to the groups and roles he has been assigned. The user, once received the information, creates a proxy certificate including assertions returned by the VOMS server. This proxy certificate is presented to the resource provider, which uses those information to decide whether granting resources or not. The user can request certificates from more than one server.

More in detail, the authorization process is accomplished following the steps below (Alfieri et al., 2005):

1. The user and the VOMS server authenticate each other using their certificates;
2. The user creates a request, signs it with his certificate, and sends it to the VOMS server;
3. The VOMS server verifies the user's identity and checks the syntactic correctness of the request;
4. The VOMS server creates a response containing the required information, signs it with its certificate, and sends it back to the user;
5. The user checks the validity of the information received;
6. The user optionally repeats this process for other VOMS servers;
7. The user creates the proxy certificate containing all the information received from the VOMS server(s); information is included in an Attribute Certificate signed by the VOMS server itself;
8. The user presents the proxy certificate to the resource provider, which decides if granting the access to resources.

3.3 Spid

SPID (Sistema Pubblico per la gestione dell'Identità Digitale) (SPID, 2014) stands for "Public System for the management of the Digital Identity"; it is a project lead by the AgID (Agenzia per l'Italia Digitale), the Italian agency in charge of the realization and diffusion of information and communication technology.

The goal is to provide a federated and centralized authentication service, by managing the registration and provisioning tools; the strong authentication implies that a certain digital identity corresponds unambiguously to a specific person. As this system has been designed primarily to facilitate the work of public administrations (but not only), the user that performs the authentication must be accurately identified. Trust is enforced by AgID, which maintains a list of entities complying with SPID specification and performs the necessary monitoring activities.

3.3.1 SPID Architecture

SPID establishes a federated environment formed by Identity Providers (IdP), Resource Providers (RP) called also Service Providers (SP), users and Attribute Authorities (AA). An IdP is an entity which stores the users digital identities, and allows users to authenticate; on the other hand, an RP is an

entity that provides services and resources. An AA releases users qualified attributes, generally requested by RP to compute permission decisions. In a federated environment, SPID allows a user to authenticate against one IdP to obtain the strictly necessary information regarding his identity, needed to access a protected service or resource offered by an RP. In this scenario authentication is delegated to the IdP rather than to the RP itself.

SPID provides three levels of authentication due to the need to comply with currently most adopted technologies as well as to keep up with the newest ones; each level complies with the international ISO/IEC DIS 29115 (ISO/IEC 29115, 2014). The first level requires username and password; the second level implies multi-factor authentication based on OTP token; the third level requires multi-factor authentication based on digital certificates.

To perform authentication, SPID can incorporate several IdPs, and each of them is an independent entity complying with SPID specifications. Thus a user digital identity is not replicated on each IdP, but is rather stored in one location. Then, when the user authenticates against the IdP that provides his identity, that authentication is accepted by any RP in the system, because RPs do not discriminate IdPs. The correct interaction sequence follows these steps:

1. the user requests access to a resource on a RP through a User Agent (UA, generally a browser);
2. the RP sends to UA an authentication request;
3. the UA is redirected to the appropriate IdP;
4. the IdP starts a challenge for credentials with the user;
5. the user provides his credentials;
6. the IdP checks for credentials correctness then returns assertions, signed by IdP itself, containing the authentication statements for the RP to the UA, otherwise if they are incorrect continues with the challenge.
7. the UA forwards the assertion created by the IdP to the RP;
8. the RP checks the validity of assertions, identifies the user and delivers the service.

4 CONSIDERATION OF IMPLEMENTATION OF IDENTITY MANAGEMENT IN THE CLOUD PLATFORM

In order to improve privacy and data protection on cloud computing, we are pioneering and studying a model in which, in order to access a resource or service supplied by a resource provider, authentication is delegated to a federated identity provider, and authorization is performed by a federated attribute authority. Specifically, we are proposing an integration of VOMS for authorization ad SPID for authentication, in the Cloud Foundry PaaS system. Such integration aims to exploit the federation environment provided by SPID, and to incorporate the qualified attribute release performed by VOMS.

4.1 VOMS Vs SPID

Although VOMS and SPID may seem similar, these are two deeply different concepts. VOMS role is to supply authorization information regarding a user in a certain domain. It provides a single service that can be executed on a single machine. SPID is a more complex system, comprises several entities that cooperate and communicate using protocols adopted by SPID. It orchestrates the interaction of IdPs, users, RPs and AAs, and also specifies the technical standards for interoperability. In this context, VOMS is one of these entities: it is essentially an AA that operates performing authorization service by issuing tokens on demand; on the other hand, SPID includes IdPs which provide authentication service.

Another relevant difference lies in the policy matters. A VOMS service do not require any accreditation, thus can be set up and made run autonomously. SPID is a system that the Italian government aims to adopt in a large scale, in order to spread the use of digital identities, thus entities must be accredited. Finally, VOMS is being used, mainly in grid computing, since several years, whereas SPID is a novel scheme targeted for the management of digital identities, thus supplies a wider and more complete view over this issues.

4.2 Integration of VOMS and SPID in Cloud Foundry

For the integration to be successful, each system must be analyzed in order to understand how it works and interacts with users or other components. In particular, we analyzed the input and output of the involved systems. The goal is to create a structure composed by SPID and VOMS that can operate and replace the Cloud Foundry build-in login system, called UAA; to accomplish this, first we need to understand its mechanisms.

The UAA (User Account and Authentication Service) is the identity management service of Cloud

Foundry. As stated in (Features of the UAA, 2012), it "is responsible for securing the platform services and providing a single sign on for web applications", and offers several features such as Centralized Identity Management, Delegation Access to Services, User Account Management and Client Application Registration.

When a user performs the login into a Cloud Foundry Resource Provider, the authentication and authorization processes are handled by this component. (UAA Server, 2012) specifies that the UAA acts both as Login Server and an OAuth 2.0 Authorization Server (OAuth 2.0, 2012), managing resources access by granting tokens which are delivered to the client application. These tokens contain, among the other, user information such as the 'user_id', the 'user_name', and the 'scope' that specifies the level of access granted by the user to the token.

Figure 1 reports and compares the input and output of each system we analyzed. This step is crucial to understand if VOMS and SPID can be integrated in Cloud Foundry, and which gaps are still to be filled.

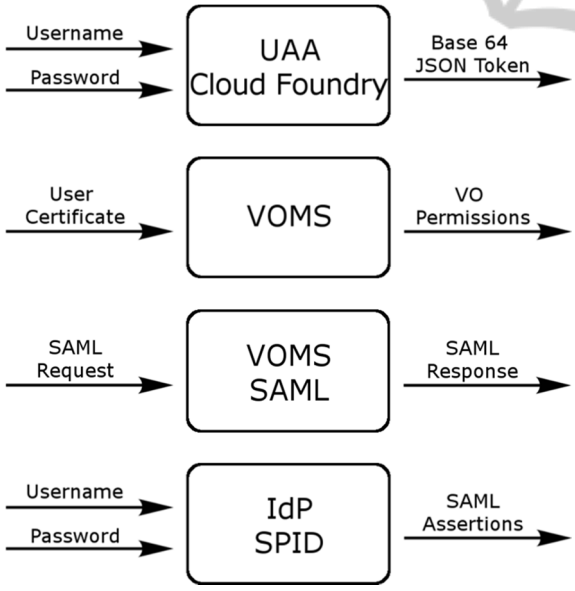

Figure 1: Comparison of the input and output requirements of UAA, VOMS and SPID.

When accessing resources on the internet, security must be ensured, especially when an authentication process is performed. SPID issues this problem by using standard SAML (SAML 2.0, 2005) message format.

SAML is an XML-based data format protocol used for authentication and authorization process. The SAML specification, accordingly with SPID requirements, defines three roles: the user, the IdP, and the RP. It specifies three components: assertions, protocol, and binding. There are three assertion types: authentication (validates the user's identity), attribute (contains specific information about the user), and authorization (specifies the user's authorizations). Then, the protocol defines the format for sending and receiving assertions, whereas binding defines how SAML messages can be mapped to SOAP messages.

On the other hand, VOMS uses certificates to address the identification of the users, needed to release the user qualified attributes; however (Venturi et al., 2008) shows how VOMS can be adapted to exchange SAML messages. VOMS runs in an own server rather than living inside the authentication server. If a HTTP server is set up, VOMS can be used to perform SAML authorization, as SAML can transport both authorization and authentication information; such SAML-based VOMS can therefore accept SAML requests containing the user identifier, for instance the user certificate, and return a SAML message composed of assertions containing user authorizations.

In this context, SPID acts as authentication service, allowing a user to provide his credential in order to authenticate in the system through an IdP, and VOMS operates as an external AA, performing authorization service.

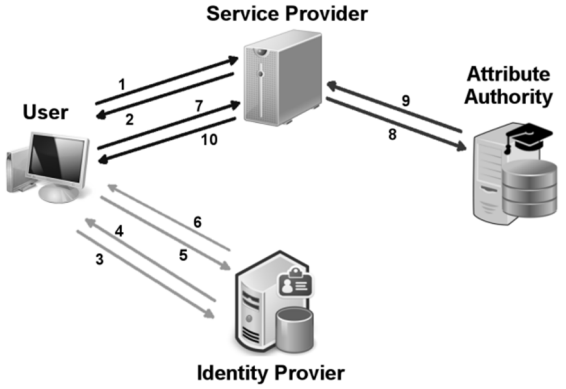

Figure 2: SPID and VOMS integration in Cloud Foundry.

Whenever a user wants to access a resource on a Cloud Foundry RP, the system performs the following steps, depicted in Figure 2, in this order:

1. the user requests access to a resource on a RP through a User Agent (UA);
2. the RP sends to UA an authentication request;
3. the UA is redirected to the corresponding IdP;
4. the IdP starts a challenge for credentials with the user;

5. the user provides his credentials;
6. the IdP checks for credentials correctness then returns assertions, signed by IdP itself, containing the authentication statements for the RP to the UA, whereas if credentials are incorrect continues with the challenge.
7. the UA forwards the assertions created by the IdP to the RP, the RP checks for validity of the assertions and identifies the user;
8. the RP sends to the VOMS an authorization request, including assertions of the authenticated user;
9. the VOMS checks for validity of assertions, and then returns a response, which includes assertions containing user authorization information, to the RP;
10. the RP checks for validity of assertions, checks if the user is authorized to access the resource and delivers it.

As already mentioned, during the communication SAML messages containing or requesting user information are sent. SAML messages are encoded using the Base64 format and compressed with a DEFLATE algorithm. In accordance with SPID guideline (SPID specifications, 2014), a SAML message, which is XML-based, must include specific tags. Among these, the most relevant is the tag that contains the assertions. An assertion is defined by the <AttributeStatement> tag in the following format:

```
<saml:AttributeStatement ...>
  <saml:Attribute...>
    <saml:AttributeValue ...>
      myAttributeValue
    </saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

Beside the benefits mentioned above, authentication (not limited to the cloud environment) still presents risks regarding security. Identity theft is one of the most concerning matters in a federated centralized environment: if a user credentials are stolen, the hacker can access all the services with the same credentials. This problem is issued by using strong authentication protocols, where information are sent over secure connections.

Moreover, a large scale deployment of such architecture requires specific software to be deployed on each entity. RPs must be able to create SAML requests, redirect users to their IdP, and receive SAML responses. IdPs must be able to receive SAML requests, perform authentication, and return SAML responses redirecting users back to the RP. SPID specifies that IdP must use non-proprietary solutions (in terms of software and hardware), in order to foster interoperability and avoid vendors technological lock-in.

Setting up a working federation based on SPID will require time to complete the specifications, test the interactions between entities and address any further technical issue. The diffusion of this framework is going to be gradual: all the Italian public administrations are expected to become SPID RP compliant within 24 months since the accreditation of the first IdP. Private entities may intentionally join the federation, once they are become SPID compliant. In addition, the number of digital identities will increase over time, together with the spreading of this technology among the territory, only if this solution will prove to be reliable, efficient and user-friendly. SPID aims to simplify and secure the user access to services, but may not succeed if such features, that are fundamental for a wide adoption of digital identities in a federation, are not ensured.

5 CONCLUSIONS

In this paper we studied a cloud computing system handling identity management issues. First we presented the state of the art of cloud technology, describing the principal advantages and downsides affecting it. The main goal was to improve the identity management in the cloud PaaS model, in particular Cloud Foundry. For this purpose, VOMS as authorization and SPID as authentication systems were chosen to be integrated. Regarding the communication, SAML was chosen as it is a protocol created to transport authentication and authorization statements. Such solution can benefit in terms of trust (since SPID provides a federate environment), usability proof (since VOMS is being used in grid since many years), and interoperability (since SAML is a open-standard format based on international XML format).

A future work may consist in setting up a system as discussed above. However, SPID is currently in a development phase, and it has not been deployed in Italian territory yet.

REFERENCES

T-Systems Enterprise, 2010, *White Paper Cloud Computing. Alternative sourcing strategy for business ICT*. T-Systems Enterprise.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., 2010. *A View of Cloud Computing*, ACM Digital Library.

NIST, 2011. *Cloud Computing Reference Architecture*. http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505. Accessed: 2015/03/17.

Dixon, J., 2014. *X as a service (XaaS):What the future of cloud computing will bring*. http://www.cloudcomputing-news.net/news/2014/aug/18/x-as-a-service-xaas-what-the-future-of-cloud-computing-will-bring. Accessed: 2015/03/17.

OpenShift, 2014. *OpenShift Online*. https://www.openshift.com/products/online. Accessed: 2015/03/17.

Salesforce, 2014. http://www.salesforce.com. Accessed: 2015/03/17.

AppScale, 2014. http://www.appscale.com. Accessed: 2015/03/17.

CloudControl, 2014. https://www.cloudcontrol.com. Accessed: 2015/03/17.

Cloud Foundry, 2014. http://cloudfoundry.org/index.html. Accessed: 2015/03/17.

Azure, 2014. http://azure.microsoft.com/en-us/services/websites. Accessed: 2015/03/17.

Amazon EC2, 2014. https://aws.amazon.com/ec2. Accessed: 2015/03/17.

OpenStack, 2014. http://www.openstack.org. Accessed: 2015/03/17.

Apache Hadoop, 2014. http://hadoop.apache.org. Accessed: 2015/03/17.

OpenNebula, 2014. http://opennebula.org. Accessed: 2015/03/17.

Heller, M., 2014. *Review: Cloud Foundry brings power and polish to PaaS*. http://www.infoworld.com/article/2608299/cloud-computing/review--cloud-foundry-brings-power-and-polish-to-paas.html. Accessed: 2015/03/17.

Alfieri, R., Cecchini, R., Ciaschini, V., Dell'Agnello, L., Frohner, A., Lorentey, K., Spataro, F., 2005. *From gridmap-file to VOMS managing authorization in a Grid environment*. FGCS.

OpendID, 2015. http://openid.net/. Accessed: 2015/03/17.

Cloud Foundry UAA, 2012. *Introducing the UAA and Security for Cloud Foundry*. http://blog.cloudfoundry.org/2012/07/23/introducing-the-uaa-and-security-for-cloud-foundry/. Accessed: 2015/03/17.

DIGIPASS, 2015. *DIGIPASS as a Service - Cloud based Authentication*. https://www.vasco.com/products/managed_services/das/digipass_as_a_service.aspx. Accessed: 2015/03/17.

PowerBroker Open, 2015. http://www.powerbrokeropen.org/. Accessed: 2015/03/17.

Conjur, 2015. *What Is Conjur?*. http://www.conjur.net/what-is-conjur/. Accessed: 2015/03/17.

OAuth 2.0, 2012. *An open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications*. http://oauth.net/2/. Accessed: 2015/03/17.

VOMS, 2014. *VOMS: Virtual Organization Membership Service*. http://toolkit.globus.org/grid_software/security/voms.php. Accessed: 2015/03/17.

Venturi, V., Riedel, M., Memon, Shi., MemonSha., Stagni, F., Schuller, B., Mallmann, D., Tweddell, B., Gianoli, A., Van denBerghe, S. et al., 2008. *Using SAML-Based VOMS for Authorizationwithin Web Services-Based UNICORE Grids*. Springer.

EMI, 2014. http://www.eu-emi.eu/. Accessed: 2015/03/17.

VDT, 2014. http://wlcg.web.cern.ch/virtual-data-toolkit. Accessed: 2015/03/17.

Alfieri, R., Cecchini, R., Ciaschini, V., Dell'Agnello, L., Frohner, A., Gianoli, A., Lorentey, K., Spataro, F., 2004. *VOMS, an Authorization System for Virtual Organizations*. Springer.

SPID, 2014. *Sistema Pubblico per la gestione dell'Identità Digitale - SPID*. http://www.agid.gov.it/agenda-digitale/infrastrutture-architetture/spid. Accessed: 2015/03/17.

ISO/IEC 29115, 2011. *ITU-T Recommendation X.1254 | International Standard ISO/IEC DIS 29115*. https://www.oasis-open.org/committees/download.php/44751/285-17Attach1.pdf. Accessed: 2015/03/17.

Features of the UAA, 2012. *High Level Features of the UAA*. http://blog.cloudfoundry.org/2012/07/24/high-level-features-of-the-uaa-2/#centralized_identity_management. Accessed: 2015/03/17.

UAA Server, 2012. *User Account and Authentication (UAA) Server*. http://docs.cloudfoundry.org/concepts/architecture/uaa.html. Accessed: 2015/03/17.

SAML 2.0, 2005. *SAML V2.0*. http://saml.xml.org/saml-specifications. Accessed: 2015/03/17.

SPID specifications, 2014. *SPID regole tecniche e modalità attuative*. http://www.agid.gov.it/sites/default/files/regole_tecniche/spid_regole_tecniche_v0_1.pdf. Accessed: 2015/03/17.