2013 IEEE International Conference on Cloud Computing Technology and Science

E-Id Authentication and Uniform Access to Cloud Storage Service Providers

João Gouveia,
Paul Andrew Crocker
IT & DI
Universidade da Beira Interior
6201-001 Covilhã, Portugal
Email: m4861|crocker@ubi.pt

Simão Melo de Sousa
LIACC & DI
Universidade da Beira Interior
6201-001 Covilhã, Portugal
Email: desousa@ubi.pt

Ricardo Azevedo
PT Inovação, SA,
Rua Eng. José Ferreira Pinto Basto
3810 - 106 Aveiro, Portugal
Email: ricardo-a-pereira@ptinovacao.pt

***Abstract*—This article describes an architecture for authentication and uniform access to protected data stored on popular Cloud Storage Service Providers. This architecture takes advantage of the OAuth authentication mechanism and the strong authentication mechanism of the National Electronic Identity (E-Id) Cards , in our case the Portuguese E-Id card or *Cartão de Cidadão* (CC). We shall present a comparison of authentication mechanisms and access to popular cloud storage providers, comparing the different authentication mechanisms *OAuth 1.0*, *OAuth 1.0a* and *OAuth 2.0*. Using the proposed architecture we have developed an implementation of this architecture that provides a uniform web based access to popular Cloud Storage Service Providers such as *Dropbox*, *Skydrive*, *Cloudpt* and *Google Drive* using the authentication mechanism of the E-Id card as a unique access token. In order to provide a uniform access to these services we shall describe the differences in the various REST APIs for the targeted providers. Finally the web application that allows users that hold E-Id cards a single point of access to their various cloud storage services will be presented.**

I. Introduction

With the exponential growth of the internet and similar increase in the number of services available to users the spread of digital identities has become endemic. In the most recent period of the history of the internet a huge effort has been made to find solutions that help solve the problems related to the explosion of the number of identities that any single user may have. The greater use of cloud services in recent years has simply added to an already known problem and it is therefore imperative for the research community to investigate new and innovative ways for users to secure their data whilst coping with multiple identities.

Identity management systems have emerged as a mechanism for users to manage their multiple identities. These systems have as main features the ability to manage individual identities, manage authentication, authorization, roles and privileges for a given service or set of services. These systems are able to provide easy access to protected data to third parties, without being required to share sensitive information such as your user name or password. The hope is that this new paradigm will fix several problems of multiplicity of identities, authentication, and confidentiality. OAuth and openID are examples of this type of systems that manage a particular form of the identity of a user. If the strong authentication of Electronic identity (E-Id) authentication could be combined with one of these protocols then a user can have guarantees of having a robust two-factor authentication system where an important component in the process of authentication is the fact of owning a physical element and knowing the PIN code and only through this mechanism can a user access services where he is registered with any number of varying identities.

The proposed architecture uses the concept of identity management systems and the concept of authentication with the Portuguese Citizen Card and applies them in Cloud environments, mainly in the most popular storage providers that support OAuth. The final implementation presented is aimed at the following providers: Dropbox, Skydrive and Cloudpt. In this way we have included and studied the behaviour of the various REST APIs offered by these providers and also shown how can we deal with all the versions of the OAuth Protocol.

The remainder of this article is organized into the following seven sections. In section II the contribution of this article will be presented. In section III related work shall be presented. In section IV the authentication aspect of the Portuguese E-ID card is explained and its integration into the proposed architecture detailed. In the following section V the OAuth protocol and its various versions are discussed. In section VI the various REST APIs of the targeted cloud storage service providers are discussed. In section VII the web application developed based on our proposed architecture and API is presented. Finally in section VIII conclusion and future work are described.

II. Contribution

The architecture proposed in this article has the objective of aggregating different cloud providers. The main contribution of this paper is the architecture and mechanisms developed so that users authenticated with a National E-Id card are given transparent access to their cloud providers by interacting with their services using the authentication and authorization provided by the OAuth protocol. Although there are already various platforms that aggregate

978-0-7695-5095-4/13 $31.00 © 2013 IEEE
DOI 10.1109/CloudCom.2013.71

IEEE computer society

cloud services, there is no platform doing this by combining the strong authentication present in national E-Id smart cards. In order to implement such an architecture it was also necessary to design and develop a new API and provide news mechanisms to interact with the different versions of the OAuth protocol that are transparent to the end user. The architecture has been implemented and tested for holders of Portuguese E-Id Cards. Cloud service providers using different authentication technologies for authenticating users means customers must deal with multiple login credentials. Another contribution of this papers is the way that this problem is partially solved by providing a single point of access secured with a smart card and authentication pin number.

III. Related Work

The management of multiple identities (username and password) is not only a nuisance, but also one of the main weaknesses of the security model of the Internet. The majority of the services has its own registration form and requires the client to register to gain access to their services. Worse than this is the fact that whenever a user registers at one of these services he/she will do so using a username and password the same or nearly identical to already existing usernames/passwords that he/she currently use, which is a bad practice but unfortunately very common.

In 2005, the first major efforts to correct this problem appeared, with the appearance of identity management systems based on community wide open standards. In the group of open standards one can include SAML, openID and OAuth as the principle standards for the single sign-on processes, authentication and authorization. Although these systems offer part of the solution, there are still some security flaws in them [9].

Pascal Urien in [16] describes a scheme almost identical to the scheme that will be applied in this article, based on the openID authentication protocol and a system of authentication based on *smartcards*. The main difference between this scheme and the one proposed in this article is based on the fact that the system is oriented to authentication with smart cards for any service that requires user authentication and supports openID, while the ultimate goal of the proposed work involves the authentication with smart cards and subsequent authorization of this entity to access protected resources. Whilst Urien is based on openID protocol because this only needs to authenticate to a specific service, Simploud [1] is based on OAuth protocol because it requires pseudo-authentication allied to the authorizing access to protected resources stored in cloud providers which support OAuth.

Also in [1] [2], Al-Sinani describes a scheme that integrates information card systems, such as *Cardspace* (this project has been discontinued) and *Higgins*, and authentication and authorization protocols, including *openID* and *OAuth*.

In this article we shall consider national E-Id smart cards, such as the Portuguese Cartão de Cidadão (CC), which allow strong authentication in public and private electronic systems. National E-ID cards such as the CC are already used as authentication tokens in diverse systems such as the Tax Services Portal, *e-Health* for health card professionals [10], and even a federated identity management system [13].

Hyuck Han et al. [11] proposed a scheme that introduces *REST* in cloud infrastructures. In [3], Giuseppina Cretella e Beniamino Di Martino, propose a methodology and techniques for the semantic analysis of this type of cloud architecture and infrastructure. Toby and Anthony Velte, and Robert Elsenpeter, in [17], analyse the cloud infrastructure, as well as all its component architectures including REST and JSON.

In this article we shall present a new architecture and implementation based on these previous works and that makes use of National Electronic identity (E-Id) cards.

IV. E-Id Cards

Electronic identity (E-Id) cards permit the holder to prove his identity, physically or digitally. This type of card contains human and machine readable information including photograph and often biometric information such as fingerprint template. The card is based on sovereign state governmental systems that include many levels of public government departments and national Public Key Infrastructure (PKI) for managing the system.

The initial concept of the Portuguese E-Id card was to merge various identification documents into a single electronic smart card and permit the maximum of interoperability between the various entities. Other objectives include permitting and enabling greater electronic interaction with the Portuguese government, simplifying the access to electronic services, providing a legal and electronic infrastructure for validating electronic documents and for the creation of innovative applications.

The Portuguese E-Id Card (CC) is a Java smart card that follows the international recommended norms to be officially recognized as an identification and travel document. This is in line with the current guidelines of the European Union, in particular with the working group for the European Citizen Card (ECC) and by standards set by the International Organization for Standardization (ISO 7501 and ISO 7810). This means that the results of this paper can be applied in the context of all other electronic identification cards that follow the same standards, requiring only the installation of the root certificates of the other cards in the server where the system is running.

The card contains information written onto the card in several formats: visual human readable information, a bar-code type machine-readable zone and also information in electronic form on the chip. The chip contains the personnel information about the citizen, such as name and address, photographic image and biometric fingerprint template. Cryptographic keys and digital certificates, each one of these has an asymmetric RSA key pair, created by the Portuguese PKI of the Portuguese Ministry of Justice are also included.

[1] https://spocs.it.ubi.pt:444/simploud/

For instance in order to use the digital authentication service and access the RSA public key the card holder must introduce his/her PIN, note that all cryptographic operations are made inside the cards chip and the private keys are never exposed outside of the card. With the CC card it is possible for the user with a computer and smart card reader to authenticate on suitably configured web sites over SSL/TLS.

A. Certificate Authority

The Certificate Authority (CA) of the Portuguese state that issues the certificates of the Portuguese E-Id card is responsible not only for the issue but also the validation and verification of them. This verification is fully supported by two specific methods: CRL (Certificate Revocation List) and OCSP (Online Certificate Status Protocol). The entity that wants to check the validity and reliability of a certificate will have to perform one of these two methods and contact appropriate service supplied by the CA.

The CA also has the ability to issue a new certificate with an expiration date included. This mechanism is important for the same certificate is not indefinitely associated with the same entity. Thus, the CA has the freedom to manage certificate validity, and if for any reason it becomes necessary to revoke a certificate CA can do it. Whenever one of these processes is unleash, the entity that owns the revoked or expired certificate have in his possession a useless certificate. To solve this problem it will be necessary that this entity make a request to obtain a new certificate. This is important because the authentication process on Simploud proceeds through authentication certificates, if they have been revoked or expired, the user will be unable to perform the authentication process.

V. OAuth Authentication Protocols

As referred to by Leiba [12], *OAuth* is an open standard authentication protocol that is part of the category of identity management systems, since these systems allow resource sharing amongst electronic services. The OAuth Core 1.0 final draft was released in 2007, the latest revision was published in [5] 2010 as the OAuth 1.0 Protocol and was published as RFC 5849. The OAuth 2.0 Authorization Framework was published in 2012.

The choice of *OAuth* as the principle authentication protocol to be used in our work and not *openID* lies in the fact that whilst *openID* is only an authentication protocol *OAuth* has a larger scope, it is capable of supplying authorization and give access to private data to third parties [14]. However one of the main reasons was that the targeted cloud platforms for our application all use this protocol for authentication and authorization.

The operating mode of *OAuth* is relatively simple and is based on the possibility that a third party, given previous access consent, may access users data stored by some service without being necessary for the user to provide his access credentials (usually designated by the username and password pair).

A simple use case is allowing a print service (consumer service) to access a users private photos that are stored by some other *service provider* without it being necessary for the consumer service to demand that the user supplies his or her credentials of the service provider.

Figure 1 illustrates the process flow of the protocol, basically divided into three principal phases: The consumer makes an authorization request to the owner of the resources. The response of the resource owner is eventually returned in the form of a token or authorization grant. Next, the consumer sends a new request that includes this token in order to be authorized to access the private data of the resource owner to the resource server. If the resource owner has permitted the sharing of data with this third party, a new definitive token, known as access token, is issued and then until this token is revoked or deleted by the resource owner or server, the consumer will have access to the resource owners protected data, with the given consent of the resource owner. Note that all requests and responses of this protocol must run over ssl.

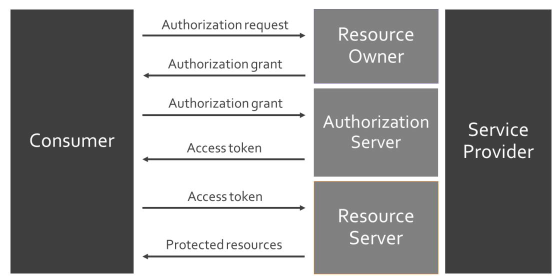

Fig. 1. *OAuth* protocol mechanism

As well as version 1.0 of the protocol two other versions exist [6] and [4].

Version 1.0a of the protocol *OAuth* resolved a specific problem due to a session fixation attack against the OAuth Request Token [7]. The attacker was able to interrupt the protocol save the authorization request *URI* (which includes the Request Token) and then if he can later convince a victim to click on a link consisting of the authorization request URI to approve access to the victims Protected Resources to the (honest) Consumer he was able to (after the victim grants approval) use the saved Request Token to complete the authorization flow, and access the victims Resources at the consumer service. This error gravelly affected all the *OAuth* mechanism, however it was detected before actual attacks were made.

Version 1.0a corrected this problem by simply introducing a verification code between the authorization request and the request to obtain the final access token in order that the server that authenticates the final consumer knows if it is the same consumer that made all the requests.

The most recent version of *OAuth* is version 2.0. It introduces new security considerations for *token* generation. The principal alterations to the protocol allow the introduction of different scopes and the concept of a *Refresh Token* [15].

The introduction of scopes in the protocol was important as it enabled a user to supply limited access to third parties, for instance to specific parts of data held by a resource server. A practical example of this situation is the Microsoft *Live API*, where a user may only wish to share data from his email account or maybe only the data held on *Skydrive*.

The concept of refresh token was important in the sense that it helps to prevent session attacks. While to obtain an access token the user only needed to send the token previously granted, for the refresh token a client ID and a client secret (OAuth parameters) are also required. Using refresh tokens the problem of session attacks was resolved, when a token expires the user only has to use a refresh request to get a new token and continue with the access granted previously. This way the danger of session attacks was limited since whenever a token expires the user only has to issue a refresh request in order to obtain a new token and continue with the access already granted.

VI. REST APIs

Recently, the *REpresentational State Transfer (REST)* architecture has been proposed as an alternative to existing web services [8]. In this era of cloud and internet services this architecture has been widely adopted, mainly due to the increased number of mobile devices and to the ever increasing number of mobile internet services.

While designing any distributed hypermedia system (modern web architecture), various factors such as scalability, simplicity, visibility, etc. should be ensured. Existing network architectural styles do ensure these factors, but there are no existing styles that integrate all these factors and desired properties. REST evolved by identifying the strengths and weakness of the existing network styles. The network styles that compose REST are *Client-Server*, *Stateless*, *Cacheable*, *Layered System*, *Code on Demand* and *Uniform Interface*.

Since this is a protocol based on HTTP requests and responses, often the services that implement REST also implement JSON format for all responses from these services. The JSON format is native from JavaScript language and is based on *Attribute : value* notation where attribute is the name with which we identify some property and value is the actual value of this property. Although born with JavaScript, this format has gained prominence, primarily due to the fact that it can work comfortably with different data structures.

For instance many of the cloud storage services feature their own REST APIs, services such as *Dropbox*, *Skydrive*, *Google Drive*, *Cloudpt*, which are subject of detailed analysis for this article.

All these providers implemented their own proprietary APIs, their own objects and JSON format. This complicates the task of constructing an API that covers all these platforms. The different versions of the OAuth protocol that these service providers use further complicates the construction of a single API for all of them. However there are parameters common to all of these APIs which are identified below:

- All the request execute over the *SSL* protocol;
- The APIs are all built around a REST architecture and the replies come in the JSON format;
- In spite of the fact that the structure of the URL requests be different for each provider, the steps that compose the process of authentication and authorization until the access to the data and files is identical;
- They all implement the basic cloud storage operations, such as *download*, *upload*, listing, sharing, copying, renaming, version recovery, move and delete.

VII. Implementation

The basic objectives for designing an architecture for E-Id Authentication and Uniform Access to Cloud Storage Service Providers was to provide users with a single point of access to all their cloud storage services and allow the possibility that third party applications could use this platform to access private data and files. In particular the architecture was designed to include an identity based encryption (IBE) system that enables users to encrypt their data on the cloud also using an IBE encryption service permits the definition of modifiable file access policies (sticky policies) when sharing data between users on such a system.

This case study was suggested and supported by *PT Inovação, SA*, included in the project *PRICE - Privacy, Reliability and Integrity in Cloud Environments* [2].

Considering the previous sections we therefore needed an architecture based on the following:

- A single point of access to multiple cloud storage service providers;
- Authentication using the strong authentication characteristic of an E-Id card;
- Authorization on the users (citizens) cloud providers using the *OAuth* protocol;
- A multi-platform API for the three cloud storage service providers that we targeted.

A Web Application [3] was developed to implement this architecture. The basic components of this application are:

- Development of a single REST *API* to interact with the three service providers;
- Authentication on the Web Site using the E-id card;
- Integrate the authentication on the Web Site with authentication and authorization on the cloud storage service providers via the *OAuth* protocol.

In order to implement authentication on the web site using the *Cartão de Cidadão* it is necessary to configure the server with the following steps:

[2] http://spocs.it.ubi.pt/price
[3] https://spocs.it.ubi.pt:444/simploud/

- Configure the Web Application server for *SSL* client certificate authentication which requires obtaining a valid and trusted server certificate and configuring the server, in our case IIS, to require client certificates;
- Configure the server to request and accept as valid certificates form the Portuguese E-Id cards;
- The server application must validate the certificate against a chain of intermediate and root certificates and of course check certificate integrity, expiry date and other parameters that can be extracted from the certificate;
- Check that the certificate has not been revoked using either CRL (*Certificate Revocation List*) or OCSP (*Online Certificate Status Protocol*).

If the authentication certificate from the smart card presents an invalid state (expired, suspended or revoked) all tokens linked to this card stored on the database will be removed to ensure the security, integrity and consistency of data. The card owner should resolve the situation online, if possible, or presentially at a trusted entity for the purpose.

The API [4] developed in this work was based on the aggregation of all considered providers and uniform interaction with its REST APIs. As explained in section VI, each provider applies their own version of *OAuth* protocol, defines its objects and JSON formats returned in responses. All these parameters lead to small changes that make the task of building this API careful but necessary.

Considering only the targeted service providers it was necessary to identify the differences between each one:

- Dropbox: Implements version 1.0 of *OAuth* and two types of different *JSON* responses, one to characterize the account details and one to store information about files and directories.
  The difference between the *file* and *directory* object is the *contents* attribute which is present in the *directory* object and represent a list of the directory content. Access to a users content is made using the *path* notation, specify a unique file location.
- Cloudpt: Implements version 1.0a of *OAuth* which implies some changes to the responses in the authentication mechanism. Version 1.0a contains an additional field, *OAuth_verifier*, an additional security code which our application must be aware of. Concerning the JSON responses, the formats actually uses by Cloudpt are identical to the previous provider, one type for file and one type for directory responses.
- Skydrive: Implements version 2.0 of *OAuth* and makes use of all the changes implemented in version 2.0 such as the concepts of refresh token and scopes.
  When the protocol is initiated its necessary to specify what are the scopes for the application. After this, and after successful authentication, the application has been explicitly authorized by the user and the consumer gets access to the designated scopes. With this version, and looking at this provider in particular, the access tokens obtained contain an expiry date, of an hour, and at the end of this time its necessary to refresh the access of the consumer to the service. Using the refresh mechanism of the *OAuth* we are able to obtain a new token without the user noticing the process.
  Concerning the JSON responses, *Microsoft Live Connect* possess objects to specify activity, album, application, audio, calendar, comment, contact, error, event,file, directory, friend, permission, photo, quota, *tag*, user and video. Although all these parameters are referred to in the *Live Connect* library only some of them are necessary for *Skydrive*.
  Our application only requires the file, directory , quota and user. The last two are necessary in order to extract the email of the user and information about how much space the user has used and how much space is still available. One last detail that distinguishes this provider to the other two is the content access, which instead of using a path type object uses an *id* property which can be obtained for a file or directory object.

[4] http://spocs.it.ubi.pt/price/api

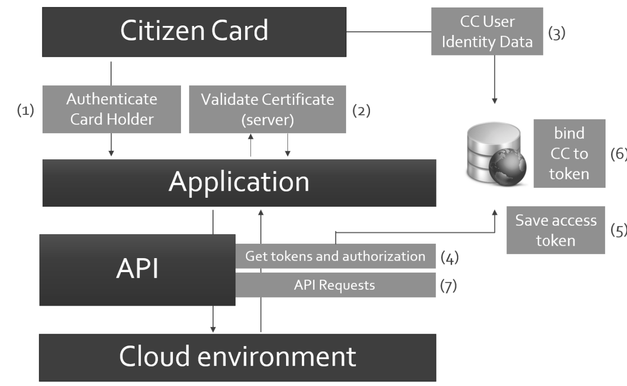

Fig. 2. Application Architecture

Figure 2 illustrates the different components in our system and indicates the different interactions between the components. The first step (1) is authentication via the smart card on the Web Application. The user must insert their smart card into the reader and enter the authentication pin to ensure the authenticity of the presented certificate. Having said this, the service sends a certificate validation request for the respective CA (2). If the answer is affirmative, the user successfully terminates authentication process with smart card, otherwise he will not be allowed to enter in the restricted area of the application.

After successful authentication the application collects some public data (name and civil id number and also issuer, serial number and public key of the public authentication certificate) and stores this data (3).

The user can then choose which service provider he wishes to access and after clicking the appropriate icon initiates the *OAuth* mechanism with this provider. This mechanism implies that the user has an account registered on the chosen provider and valid login to allow Simploud's access to their protected data. If the process is successful a token is returned which guarantees access to the protected data of the user (4). This token is stored (5) by the application together with the users public certificate (6). This means a user with the same CC Card on a subsequent return to the application does not need to initiate a new *OAuth* authorization process, at least while the access token is valid and has not been revoked or eliminated by the user. When a user with a CC is in possession of an access token the user is able to access and manipulate all the data that is associated with the obtained token (7).

VIII. Conclusions

Identity management systems are one factor in helping to solve the problem of the propagation of multiple identities of the same user and helps to make it possible to share data between different services without necessarily implying the exchange of authentication credentials such as username and password. However the propagation of multiple services all implementing different varieties of underlying authentication and authorization protocols as well as servicing information via REST in varying formats makes it difficult to design a system where the user is able to access these services from a single entry point. Also competing authentication schemes such as using smart cards or username and passwords often confuse the users.

For those users who desire strong authentication and wish to use their E-Id cards to access ever more diverse services our application makes this possible. The architecture, APIs and developed application described in this article have shown how it's possible to combine the strong authentication of E-Id *smartcards* with authentication in services that are based on the *OAuth* protocol and also aggregate these services into a single uniform access which helps to mitigate the usual problems of authentication via multiple identities in these different services.

References

[1] Haitham S. Al-Sinani. *Browser Extension-based Interoperation Between OAuth and Information Card-based Systems.* Royal Holloway, University of London, September 2011.

[2] Haitham S. Al-Sinani. Integrating oauth with information card systems. In *IAS*, pages 198–203, 2011.

[3] Giuseppina Cretella and Beniamino Di Martino. Semantic web annotation and representation of cloud apis. In *Proceedings of the 2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, EIDWT '12, pages 31–37, Washington, DC, USA, 2012. IEEE Computer Society.

[4] Ed. D. Hardt. The oauth 2.0 authorization framework. http://tools.ietf.org/html/rfc6749, October 2012.

[5] Ed. E. Hammer-Lahav. The oauth 1.0 protocol. http://tools.ietf.org/html/rfc5849, April 2010.

[6] et al. Eran Hammer-Lahav. Oauth core 1.0 revision a. http://OAuth.net/core/1.0a/, June 2009.

[7] et al. Eran Hammer-Lahav. Oauth security advisory: 2009.1. http://OAuth.net/advisories/2009-1/, April 2009.

[8] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University Of California, 2000.

[9] Eghbal Ghazizadeh, Jamalul-lail Ab Manan, Mazdak Zamani, and Abolghasem Pashang. A survey on security issues of federated identity in the cloud computing. In *Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, CLOUDCOM '12, pages 532–565, Washington, DC, USA, 2012. IEEE Computer Society.

[10] Helder Gomes, João Paulo Cunha, and André Zúquete. Authentication architecture for ehealth professionals. In *Proceedings of the 2007 OTM confederated international conference on On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part II*, OTM'07, pages 1583–1600, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] Hyuck Han, Shingyu Kim, Hyungsoo Jung, Heon Y. Yeom, Changho Yoon, Jongwon Park, and Yongwoo Lee. A restful approach to the management of cloud infrastructure. In *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, CLOUD '09, pages 139–142, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Barry Leiba. Oauth web authorization protocol. *IEEE Internet Computing*, 16(1):74–77, 2012.

[13] Frank Pimenta, Claudio Teixeira, and Joaquim Sousa Pinto. Globalid - privacy concerns on a federated identity provider associated with the users' national citizen's card. In *Proceedings of the 2010 Third International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services*, CENTRIC '10, pages 16–21, Washington, DC, USA, 2010. IEEE Computer Society.

[14] Laurie Rae, David Recordon, and Chris Messina. *OpenID: the Definitive Guide.* Oreilly & Associates Inc, 1st edition, 2012.

[15] M. McGloin P. Hunt T. Lodderstedt, Ed. Oauth 2.0 threat model and security considerations. http://tools.ietf.org/html/rfc6819, January 2013.

[16] Pascal Urien. An openid provider based on ssl smart cards. In *Proceedings of the 7th IEEE conference on Consumer communications and networking conference*, CCNC'10, pages 444–445, Piscataway, NJ, USA, 2010. IEEE Press.

[17] Toby Velte, Anthony Velte, and Robert Elsenpeter. *Cloud Computing, A Practical Approach.* McGraw-Hill, Inc., New York, NY, USA, 1 edition, 2010.