Peer-to-Peer Delegation for Accessing Web Services

Michele Tomaiuolo, Paola Turci
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma, Parma, Italy
{michele.tomaiuolo, paola.turci}@unipr.it

Abstract — Hierarchical collaborations between cooperative, rational agents are quite naturally achieved through goal delegation. In the context of a service-oriented architecture, agents responsible for workflow management can subdivide their goals in sub-goals, generate a utility function from each sub-goal and set up a negotiation process with the agents associated to one or more Web services and responsible for the interaction with them. However, such delegations cannot come into effect unless they are associated with a corresponding delegation of privileges, which are needed to access some resources and achieve desired goals. In this paper we present a security mechanism for SOAP-style and REST-style Web services that allows the distribution of the delegation of access rights among different services and clients.

Security; delegation; authorization; Web services.

I. INTRODUCTION

A number of architectures and systems are being proposed as a ground for improved interoperability among diverse systems, mainly exploiting the idea of a service-oriented architecture. There are two preferred ways of realizing a service-oriented architecture based on Web services, i.e. SOAP-style and REST-style. REST Web Services have been enjoying increasing popularity in the last years. The rationale, upon which REST is based, is quite simple, i.e. the use of long-established Web technologies instead of new standard specifications. In particular, REST-style Web services are a design paradigm in which web services are viewed as resources and can be identified by their URLs. On the other hand, SOAP-style Web services may be more appropriate when a formal contract must be established to describe the interface offered by web services or when developers must address complex nonfunctional requirements. Therefore, depending on the particular application scenario, one has to decide the best approach to use.

The adoption of a service-oriented paradigm based on Web services has definitely many benefits, but security is still a great concern. A lot of efforts, by various standards groups such as W3C, WS-I, OASIS, etc. have been devoted to web service security standards in recent years. A basic way of achieving security is relying on a secure transport layer, typically HTTPS and TLS. However, a message-level security is required in the case of architectures in which intermediaries can manipulate messages on their way. This was the rationale for the definition of new specifications, such as WS-Security [23]. WS-Security, by using the XML-signature and XML-encryption specifications, defines a standard way to secure SOAP messages, independent from the underlying transport protocol. As far as the REST-style is concerned, the security model is not as highly-developed as the security model for SOAP. Nevertheless, in both cases the focus is on individual Web services and the access issues in composed services or in the case of the presence of intermediaries between the requester and the resources have not been taken into consideration. The problem becomes more complex when the use of workflows involves many layers of services.

Let us consider, for instance, a heterogeneous society of agents, where different members have different internal complexity. In such a heterogeneous society, hierarchical collaboration between reasoning capable agents is achieved mainly through goal delegation. From the perspective of this example, the most interesting types of agents composing the society could be: the WS-manager agent and the workflow manager agent. Each WS-manager agent is associated to one or more Web services and is responsible for the interaction with them. Workflow managers have the goal of supporting users in the process of building workflows, composing external Web services and monitoring their execution. The workflow manager agent assumes the role of the delegate agent in a goal delegation protocol, subdivides its goal in sub-goals, generates a utility function from each sub-goal and sets up a negotiation process with the WS- manager agents. In such a scenario, these delegations cannot come into effect unless they are associated with a corresponding delegation of privileges, which are needed to access some resources and complete delegated tasks, or achieve desired goals.

In this paper we present a security mechanism for SOAP-style and REST-style Web services that allows the distribution of the delegation of access rights among different services and clients. This paper is organized as follows. In Section 2 we give background information on WS-* and RESTful Web services with particular attention to security issues. The third section briefly discusses the related work. Then, in Section 4, the basics of peer-to-peer delegation are introduced and in the following section a generic library and some services implementing those basic mechanisms are presented. Finally, in the last section, some conclusions are drawn about this work.

II. SERVICES ON THE WEB: SECURITY ISSUES

REST-style and SOAP-style Web services are not mutually exclusive nor is one better than the other. Both are valid approaches to solving real problems, each with its strengths

and weaknesses. The choice of which approach to use should be based on the characteristics of the application being developed.

At a fundamental level the difference between REST-style and SOAP-style Web service is ascribable to the difference between resource-oriented and activity-oriented services. Resource-oriented services focus on a collection of resources upon which a set of basic operations can be performed. The operations that can be performed are defined by the HTTP specification, i.e. retrieving, creating, modifying and deleting resources. In other words, this means working directly with the HTTP interface down at the transport layer, rather than addressing system-specific interfaces and using messages for sending the invocation details of Web services. On the other hand, an activity-oriented service focuses on actions that one can perform. Actions are the center of the attention, as opposed to resource-oriented services where operations that can be performed remain basically constant regardless of the type of resources. After all, in the REST perspective the Web is seen as the means for publishing globally accessible resources and for delivering services to clients, whereas in the SOAP context the HTTP protocol is only exploited as a binding transport protocol and the selection of the operation to be performed is specifyed in the SOAP message. Such differences have obvious consequences on the way security is implemented in the two approaches

The Web service specifications (WS-*), taking advantage the SOAP header as an extensible container for message metadata, provides developers with a set of optional specifications including those which cover the security issues. The WS-* specifications are designed in order to be composed with each other. WS-Security provides a level of abstraction which allows different systems, using different security technologies, to communicate securely using SOAP in a way which is independent from the underlying transport protocol. This level of abstraction allows developers to use existing security infrastructure but also to incorporate new security technologies. It provides a set of security features, built on established industry standards for authentication and XML encryption and signing, which supports the definition of security tokens inside SOAP messages, the use of XML Security specifications to encrypt and sign these tokens and to sign and encrypt other parts of a SOAP message. Recent specifications provide further SOAP-level security mechanisms. WS-SecureConversation defines security contexts, which can be used to secure sessions between two parties. WS-Trust specifies how security contexts are issued and obtained. It includes methods to issue, validate, renew and forwarding security tokens, to exchange policies and trust relationships between different parties. Finally, WS-Policy allows organizations, exposing Web Services, to specify the security requirements of their services. This specification provides a general purpose model and the corresponding syntax to describe the requirements and constraints of a Web service as policies, using policy assertions.

No framework for advanced security, equivalent to that provided by WS-*, has been proposed for REST. The simplicity of REST if compared with SOAP and WS-* stack is real until it is carried out an ad hoc integration over the Web, but if advanced functionalities, as those delivered by WS-*, are needed, it is not so simple to extend REST-style Web services in order to support them in an interoperable manner. For less demanding scenarios, both REST and SOAP styles take advantage of the basic guarantees provided by protocols such as HTTPS and TLS.

III. Related Work

The Web Services access control is already becoming an important topic of many recent researches. The various security standards proposed and most of the studies carried out in the context of Web services focus mainly on the access control policies for single web services [4][5][6]. In particular, in [5] the authors address the problem of securing sequences of SOAP messages exchanged between web services and their clients. By constructing formal models they investigate the security guarantees offered by the specifications WS-Trust and WS-SecureConversation, which provide mechanisms allowing communicating parties to establish shared security contexts and to use them to secure SOAP-based sessions.

A few research works have dealt with security issues related to composed services.

She et al. [29] propose a delegation-based security model to address problems such as how much privilege to delegate, how to confirm cross-domain delegation, how to delegate additional privilege. The proposed model extends the basic security models and supports flexible delegation and evaluation-based access control. But all web services participating in this composition have to agree on a single token-based authorization mechanism, i.e. a hierarchical access control framework is provided.

In [9] a delegation framework which allows delegation of access rights in multi-domain service compositions is presented. The approach is based on an abstraction layer, called abstract delegation, which harmonises the management of heterogeneous access control mechanisms and offers a unified user experience hiding the details of different access control mechanisms. Our approach differs from this because we consider each service or resource as a trust domain based on a certificate chain access control mechanism.

IV. Delegation

The traditional approach for inter-domain security is based on centralized or hierarchical certification authorities and public directories of names [13][14][16]. In contrast with this hierarchical approach, other solutions are possible, where the owner of local resources is considered as the ultimate source of trust about them, and he is provided with means to carefully administer the flow of delegated permissions [7][8][18]. Trust management principles argue that no a-priori trusted parties should be supposed to exist in the system, as this would imply some “obligated choice” of trust for the user, and without choice, there is no real trust. Moreover, the presence of some third party as a globally trusted entity implies that all systems participating in the global environment have to equally trust it.

Nowadays, new technologies, in the form of protocols and certificate representations, are gaining momentum. They allow

a different approach towards security in global environments, an approach which is paradoxically founded on the concept of "locality". Federation of already deployed security systems is considered the key to building global security infrastructures. In this way, users are not obliged to adopt some out of the box solution for their particular security issues, to rebuild the whole system or to make it dependent upon some global authority, in order to gain interoperability with others.

Instead they are provided with means to manage the trust relations they build with other entities operating in the same, global environment. In the same manner as people collaborate in the real world, systems are being made interoperable in the virtual world. Cooperation and agreements among companies and institutions are making virtual organizations both a reality and a necessity. But they will never be very successful if existing technologies will not match their needs.

The Simple Digital Security Infrastructure (SDSI) [1][15][28], which eventually became part of the SPKI proposal [10], showed that local names could not only be used on a local scale, but also in a global, Internet-wide, environment. In fact local names, defined by a principal, can be guaranteed to be unique and valid in its namespace, only. However, local names can be made global, if they are prefixed with the public key (i.e. the principal) defining them. There's no limitation to the number of subjects (keys or other names) which can be made valid meanings for a name. So in the end, a name certificate defines a named group of principals. Some authors interpret these named groups of principals as distributed roles [19][20][21]. The case where a group contains other groups is interpreted as a role-subroles relation. While the SPKI proposal was based on s-expressions for representing certificates, the theory on which the proposal is based doesn't force a particular representation.

Recently, the SAML language emerged as the standard for representing security assertions [24]. Since the specifications allow a quite wide range of assertion types to be issued, it is also possible to use SAML to represent delegation certificates based on trust management principles and on the SPKI theory.

The generic structure of a SAML assertion makes evident it is very similar to what is usually called a "digital certificate". Like in every other certificate, an issuer attests some properties about a subject, digitally signing the document to prove its authenticity and to avoid tampering. Conditions can be added to limit the validity of the certificate. As usual, a time window can be defined. Moreover, it can be limited to a particular audience or to a one-time use. Conditions can also be put on the use of the certificate by proxies who want to sign more assertions on its basis.

Being designed to allow interoperability among very different security systems, SAML offers a variety of schemes to format security assertions. One interesting possibility is to use a SubjectConfirmation object to represent a subject directly by its public key, which resembles the basic concepts of SPKI, where, at the end, principals "are" always public keys.

The possibility to link local namespaces in a global scale, paves the way for a new paradigm for distributed security. This paradigm is sometimes named dRBAC, distributed Role-based Access Control. In particular, some authors [12] argue that dRBAC should add some new features to previous approaches:

• Third-Party delegations allow some entities to delegate roles in different namespaces. This mechanism, related to the "speaks for" relationship in the Taos system, does not add any new functionality, as the same results can be obtained using anonymous intermediate roles, but improves the expressiveness and manageability of the system.

• Valued attributes allow to add attributes and corresponding numeric values to roles. This way, access rights for sensible resources can be modulated according to some attributes. The same result could be obtained by defining different roles for different levels of access rights, but this would multiply the number of needed roles.

• Continuous monitoring allows to verify the actuality of trust relationships. Typically, this feature is based on a publish/subscribe protocol to advertise the status updates of relevant credentials, which can be either revocable or short-lived.

V. Implementation and Discussion

In the following the implementation of security mechanisms for web services, based on peer-to-peer delegation, will be presented. In particular, a generic library has been developed, which allows issuing and verifying chains of delegation certificates and thus allows associating a particular request with some roles and permissions. Furthermore, a SOAP based security service has been developed, responsible for allowing the creation of a security session on a platform, so that a client can send his chain of delegation certificates just once, and then possibly access the services provided on that platform. A quite similar service has also been developed according to the RESTful paradigm. Finally, an extension of our delegation framework is proposed, with the aim of taking into account the OpenID protocol.

A. Delegation library

The first step to develop a security infrastructure for web services consisted in the realization of a software library implementing core functionalities, i.e. allowing the creation and validation of delegation certificates and certificate chains. This software library can be used to manipulate SAML and XACML structures. Unfortunately, probably due to the relative novelty of relevant standards (especially for their latest versions), the software park is not particularly vast.

With regards to SAML, the choice falls on the OpenSAML library. In fact, while still being in a development phase, it is the only one supporting all functionalities of SAML 2.0 and, above all, allowing the definition of new classes with relative simplicity. Extensibility is in fact particularly important, in our case, to realize a "glue" level between SAML and XACML [25], embodied by the XACMLPolicyStatement element.

About XACML, instead, the choice of Sun's XACML Implementation was obliged, in practice, as it is the only valid open source tool to deal with the language.

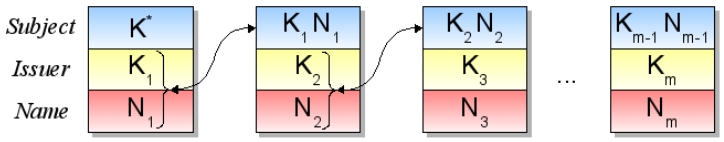

Figure 1. Delegation chain

Then, it was decided to give a standard structure to our library, realizing its API like a Java security provider. The Java Cryptographic Architecture (JCA) foresees in fact the possibility to realize packages, called security provider, which provide JDK with a concrete implementation of a subset of Java cryptograohic functionalities. For developers wanting to use the library, the main advantage of this choice is the availability of a set of API with a well known and collaudated structure. Moreover, this will allow the use of certificates and paths which will be realized with normal Java API, without duplicating their functionalities. In fact, in principle any component (also external ones), operating on a Java certificate, will be able to operate on a certificate of the new library, too.

To realize an extension of the Cryptographic Architecture, first of all it was necessary to extend Java basic data types, which in our case are represented by certificates and paths; then engine classes had to be realized, which specify algorithms to be implemented. Finally, a master class for the provider had to be implemented, which is necessary to register new classes and allow them to be used by Java.

To represent certificates, Java cryptographic APIs define an abstact class: Certificate. Within it, all basic methods to manage public key certificates can be found. Extending this class, an abstract class has been realized, containing the common methods of its derived classes, representing name certificates and authorization certificates.

An algorithm to evaluate the correctness of a certificate chain is described in the original SPKI proposal. To this aim, a subclass of CertPathValidator had to be developed, implementing this validation algorithm (see Fig.1). Parameters of the validation process are represented as ValidatorParameters objects, containing the list of keys trusted by the principal operating the verification, and possibly additional parameters.

A further operation to be offered by the library is that of validating a request to access a local resource. The request itself is represented by an instance of the AuthorizationRequest interface. Users of the library can provide different implementations of the interface, according to their needs.

Apart from the request, the algorithm with the list of authorization certificates to use and the list of trusted keys needed during the certificate verification process must be provided. Finally, in the case some additional conditions exist, it could be necessary to specify additional parameters for the verification process.

The validation happens through the creation of a Policy Decision Point (PDP). The Sun's XACML library provide the methods for creating such a decision block. However, to be able to obtain all needed policies, to validate the request, the PDP class of XACML uses various finder modules allowing to retrieve information. It was thus necessary to develop a finder module, called AuthzPolicyFinderModule, which is in charge of retrieving policies from authorization certificates provided as parameters.

During the process of creation of a PDP it is possible to insert additional finder modules. Such modules can be specified in the phase of construction of the AuthorizationEvaluator object and allow to extend the object's capabilities to search for information. Moreover, this way it is possible to provide the validation module with a series of local policies which are not stored within SPKI authorization certificates.

The final result of the operation is a list of AuthorizationResponse objects, one for each resource which was asked to be accessed. Each instance contains in its structure an identifier of the resource which it refers to, a decision value and a status code.

B. SOAP services

The objective of this first sub-project was to create a security mechanism for web services based on the SOAP protocol. The mechanism had to allow the distributed delegation of access rights among different services and clients. Instead of attaching a certificate chain to each service request, a generic security service was designed. This service had to accept and verify a certificated chain attached to a signed authentication request. After a successful authentication, the client had to be associated with a security session, which it could then mention when trying to access services on a particular platform. The session id had to be obtained according to the WS-Trust specifications and it had to be used as a meaningful security token to be associated with WS-Security enriched messages.

The sub-project has been implemented using the Axis framework, and resulted in a generic authentication service, a dummy service which needs proper authorization to be accessed, and a prototype client (see Fig.2).

All three parties are associated with their own couple of private and public keys, and can manage chains of delegation certificates encoded as SAML assertions. Moreover all parties leverage Rampart to generate signed SOAP messages conforming to WS-Security specifications.

Thus, the project effectively uses a number of technologies which have already been tested, and can work together to realize more complex scenarios than the ones foreseen in their specifications.

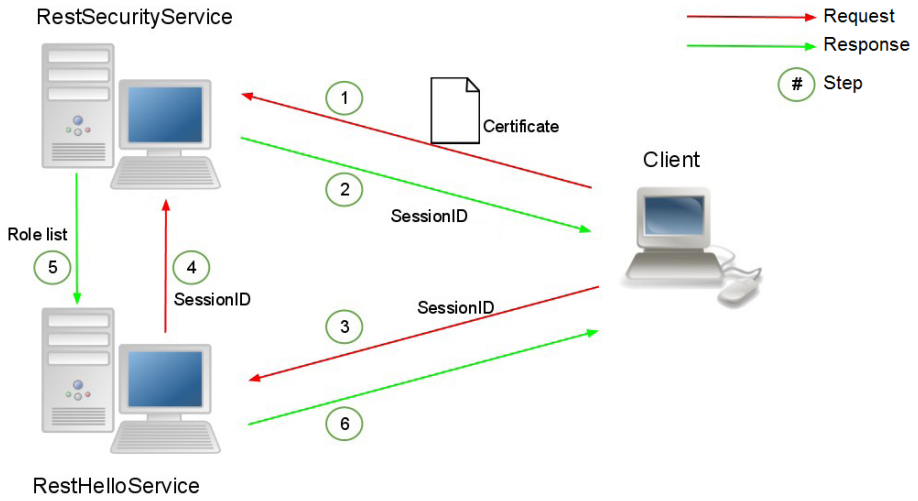

Figure 2. Using delegation certificates for accessing Web services

The realized security Web Service can effectively handle authentication requests, i.e. verify the message signature, verify the chain of delegation certificates, and eventually generate a security session and return a session identifier to the client. It is not yet associated with an explicit security policy, as defined by the WS-Policy specifications. Instead, the client has to possess a-priori knowledge of security requirements.

The client is built as an example and illustrates all the steps that a user application has to complete, to use the delegation mechanism.

The dummy service, finally, has an associated Axis Handler to manage the session abstraction and verify the proper authorization before granting access to the service. Under the hood, the handler contacts the security service to receive a list of distributed roles associated with the public key and session id of the client, and then it uses an XACML policy to verify the association of the roles with the required permissions.

C. *RESTful services*

This sub-project replies in large part the previous one, but in a RESTful environment. The main actors are still a client, a security service, to handle authentication requests and sessions, and a dummy service, which exploits the security service to implement its access control mechanisms (see Fig.2).

Some differences, though, derive directly from the different stack of involved protocols. The RESTful approach is much simplified with respect to the SOAP approach. Messages are plain HTTP messages, and security is limited to TLS and HTTPS. In our scenario, we also introduced some variations, to exploit the specific features of the REST environment. First of all, the client was reduced to a plain web browser, which generates all requests and takes care of the cryptography. For this purpose, we installed a private/public key pair (encoded into self-signed PKCS#12 certificates) in Firefox and enabled the not very popular policy of mutual authentication, allowed by HTTPS. Another difference we introduced was to send the chain of certificates not directly in the request body, but as urls of signed SAML documents available as resources on the web. This way, the composition of the request is simplified for the user, and moreover this opens up the possibility of renewing the delegation certificates automatically, and making the most recent issue available in a well known location.

The framework used, for the development of the RESTful web services themselves, is Jersey. The internal functioning of the services remains the same as in the previous sub-project, but the APIs change for adhering to the chosen paradigm. For implementing the dummy service, a Filter was created, which takes care of contacting the security service and matching the acknowledged roles with the required permissions.

D. *Integration with OpenID*

Installing certificates in a browser is not always possible or desirable. It may be practical for accessing services from a personal device, but this would limit the integration of the application in the web at large.

OpenID [26] is a decentralized digital identity system, in which any user's online identity is given by URL (such as for a blog or a home page) and can be verified by any server running the protocol.

The main motivation for OpenID is to avoid Internet users, in particular users of blogs, wikis and forums, to create and manage a new account for every site they intend to contribute in. Instead, on OpenID enabled sites, users only need to provide their home url, so that the authentication process can be completed with their own identity provider.

A limitation which has been often highlighted, is that OpenID does not allow to describe the authentication and login mechanism explicitly. When the knowledge of used mechanism is needed by a relying party, before accepting a remote authentication notification, it must be obtained by other means. This is the case of access to sensitive data, for example in the context of e-banking applications, which require the use of strong authentication mechanisms. To solve this issue, the

integration of OpenID with SAML has been proposed. In this case, SAML can be used to provide explicit information about the authentication context.

Another limitation is associated with the very idea of completely "open" authentication, as in fact a malicious user or software agent can provide its own authentication server. Thus the whole mechanism does not improve security in any way. As a consequence, "open" authentication soon turned into federation among authentication domains, using white or black lists of known OpenID providers.

Yet, the main problem is that, even if integrated this way with SAML or used into a federation of security domains, OpenID still remains focused on authentication, thus its usefulness and applicability is confined to very simple applications, where trust relationships are not built among users, with delegation of access rights, but instead based on federation of identity providers. However, in the generic context of service composition, above all in open peer to peer networks, identity information alone (especially if it is provided by some remote host) is not sufficient to take decisions whether to grant access to a local resource or a service, or not.

The next sub-project, which we are working on, in the context of this research deals with the integration of OpenID into a trust management environment. The goal is to substitute, in the last ring of a delegation chain, the public key with an OpenID url to authenticate the final user of the service. In this way, on the one hand, the remote identity is associated with distributed roles and thus to local access rights. On the other hand, an identity provider is trusted when it is included into a chain of delegation, thus eventually allowing to avoid the global white and black lists of identity providers.

This sub-project will have to overcome some problems related to the secure communication of the credentials, but above all it will have to deal with (and probably live with) important differences between the paradigms: one completely decentralized, the other one based on a unique hierarchy of names (urls) and on globally trusted third parties to assure the secure communication among all peers.

VI. Conclusion

While the traditional approach for inter-domain security is based on centralized or hierarchical certification authorities and public directories of names, new solutions are appearing. Trust management systems do not assume, a-priori, the existence of some globally trusted parties. A number of emerging technologies, including SAML and XACML, can enable this kind of solutions in the context of web services. This work analyzed the use of peer-to-peer delegation mechanisms in the context of SOAP services and RESTful services, using the relevant standards defined for the two different approaches. The results of this work include a generic library for issuing and verifying delegations chains, a security service with a SOAP interface, a security service with a RESTful interface, plus prototype components representing clients and final services to be deployed in an open environment.

References

[1] Abadi, M. (1998). On SDSI's Linkd Local Name Spaces. Journal of Computer Security, 6 (1-2), 3-21.

[2] Anderson, A., Lockhart, H. (2004). SAML 2.0 profile of XACML. Retrieved April 20, 2009, from http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf

[3] Aura, T. (1998). On the structure of delegation networks. In Proc. 11th IEEE Computer Security Foundations Workshop (pp. 14-26). IEEE Computer Society Press.

[4] Bertino E., Squicciarini A. C., Paloscia I., and Martino L. (2006). Ws-AC: a fine grained access control system for web services. World Wide Web. Vol. 9. No. 2. pp. 143-171.

[5] Bhargavan, K., C. Fournet, C., Gordon, A.D., Corin, R. (2007). Secure sessions for web services. ACM Transactions on Information and System Security. Vol. 10. Issue 12. 2007.

[6] Bhatti R., Joshi J. B. D., Bertino E., Ghafoor A., (2003). Access Control in Dynamic XML-based Web-Services with XRBAC, In proceedings of The First International Conference on Web Services, Las Vegas.

[7] Blaze, M. , Feigenbaum, J., Lacy, J. (1996). Decentralized trust management. In Proc. of the 17th Symposium on Security and Privacy (pp. 164-173). IEEE Computer Society Press.

[8] Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A. (1999). The KeyNote Trust-Management System Version 2. IETF RFC 2704, September 1999. Retrieved April 20, 2009, from http://www.ietf.org/rfc/rfc2704.txt

[9] Bussard, L., Nano, A., Pinsdorf, U. (2009). Delegation of access rights in multi-domain service compositions. IDIS Journal (Identity in the Information Society), volume 2, n. 2, p. 137-154. Springer Netherlands.

[10] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T. (1999). SPKI certificate theory. IETF RFC 2693, September 1999. Retrieved April 20, 2009, from http://www.ietf.org/rfc/rfc2693.txt

[11] Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid-enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 15(3), 200-222.

[12] Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V. (2002). dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments. icdcs, pp.411, 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02), 2002.

[13] Gutmann, P. (2004). How to build a PKI that works. 3rd Annual PKI R&D Workshop. NIST, Gaithersburg MD. April 12-14, 2004.

[14] Gutmann, P. (2000). X.509 Style Guide. Retrieved April 20, 2009, from http://www.cs.auckland.ac.nz/~pgut001/pubs/x509guide.txt

[15] Halpern, J. van der Meyden, R. (1999). A Logic for SDSI's Linked Local Name Spaces. In Proc. 12th IEEE Computer Security Foundations Workshop (pp.111-122).

[16] Housley, R., Polk, W., Ford, W., Solo, D. (2002). Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC 3280, April 2002. Retrieved April 20, 2009, from http://www.ietf.org/rfc/rfc3280.txt

[17] Khare, R., Rifkin, A. (1997). Weaving a web of trust. World Wide Web Journal, 2 (3), 77-112.

[18] Lewis, J. Reinventing PKI: Federated Identity and the Path to Practical Public Key Security. 1 March 2003. Retrieved April 20, 2009, from http://www.burtongroup.com/

[19] Li, N., Grosof, B. N., Feigenbaum, J. (2003). Delegation logic: a logic-based approach to distributed authorization. ACM Transactions on Information and System Security. Vol. 6. No. 1. pp. 128-171. 2003.

[20] Li, N. (2000). Local names in SPKI/SDSI. In Proc. 13th IEEE Computer Security Foundations Workshop (pp. 2-15). IEEE Computer Society Press.

[21] Li, N., Grosof, B. (2000). A practically implementable and tractable delegation logic. Proc. 2000 IEEE Symposium on Security and Privacy (pp. 29-44). IEEE Computer Society Press.

[22] Moses, T. (2005). eXtensible Access Control Markup Language (XACML) Version 2.0. Retrieved April 20, 2009, from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[23] OASIS. Web Services Security: SOAP Message Security 1.1. http://www.oasis-open.org/committees/download.php/16790/wssv1.1-spec-os-SOAPMessageSecurity.pdf. Feb., 2006.

[24] OASIS Security Services (SAML) TC. http:// www.oasis-open.org/committees/security/.

[25] OASIS eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/xacml/.

[26] OpenID (2007). OpenID Authentication 2.0, December 5, 2007. Retrieved April 20, 2009, from http://openid.net/specs/openid-authentication-2_0.html

[27] Ragouzis, N., Hughes, J., Philpott, R., Maler, E., Madsen, P., Scavo, T. (2008). Security Assertion Markup Language (SAML) V2.0 Technical Overview. Retrieved April 20, 2009, from http://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf

[28] Rivest, R.L., Lampson, B. (1996). SDSI - A Simple Distributed Security Infrastructure. September 15, 1996. Retrieved April 20, 2009, from http://people.csail.mit.edu/rivest/sdsi11.html

[29] She, W, Thuraisingham, B, Yen, I-L. (2007). Delegation-based security model for web services. In: Proceedings of 10th IEEE High Assurance Systems Engineering Symposium (HASE '07). IEEE Computer Society. p. 82–91. ISBN:978-0-7695-3043-7.

[30] Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S., Siebenlist, F. (2004): X.509 Proxy Certificates for Dynamic Delegation. Proceedings of the 3rd Annual PKI R&D Workshop. Gaithersburg MD, USA, NIST Technical Publications.