Policy Management Architecture Based on Provisioning Model and Authorization Certificates

Arlindo L. Marcon Junior, Altair O. Santin, Luiz A. de Paula Lima Jr, Maicon Stihler
Pontifical Catholic University of Paraná – Graduate Program in Computer Science
Curitiba – Paraná – Brazil
(almjr, santin, laplima, stihler)@ppgia.pucpr.br

ABSTRACT
The unified management of user rights and access control policies in a corporation with many units is not easy to implement. Moreover, most of the distributed access control systems are complex and heterogeneous, making it hard to maintain a unified control over all fine grained policies employed by each unit. This paper proposes a unified administration of policies for corporation environments by applying a management scheme based on authorization certificates. These certificates allow the derivation of new fine grained policies in the domain of each unit, assuring that no corporation policies will be violated. These new policies update automatically the corporation repository, preserving the unified management of user rights, and then update the corresponding policy repository of each unit. Our proposal provides a real loosely coupled policy management scheme using a serverless public key infrastructure and the Web Services technology. The prototype shows the proposal viability.

Categories and Subject Descriptors
D.4.6 (Operating Systems): Security and Protection – *access controls, authentication, cryptographic controls.*

General Terms
Management, Security.

Keywords
Policy Management, Policy Provisioning, Authorization Certificates and Web Services Security.

1. INTRODUCTION
The expansion of communication networks allows geographically separated corporations to envisage both the integration of their computer systems and the provision of services through the Internet. Nowadays, interactions involving entities such as corporations, cooperating partners or customers are becoming everyday occurrences. In such a scenario, traditional assumptions for establishing and managing rights and enforcing access control rules are no longer viable. The entities need to authenticate and trust each other in order to exchange sensitive information and to share and access resources.

SAC'09, March 8-12, 2009, Honolulu, Hawaii, U.S.A.

Therefore, in order to cope with the heterogeneity of platforms some standardization may be required to deal with information exchange, management of rights, establishment of trust relationships and so on.

Web Services (WS) have been quickly adopted by the corporate world. However, WS inherit policy control mechanisms from the traditional security architectures. Therefore, they depend strongly on providers (servers) in the corporation's domain and on centralized access control mechanisms. Additionally, relationships among the distributed entities depend on a trusted third party – normally a unique domain authentication service.

Although global administration of corporation policies facilitates the imposition of rules to the distributed environment, it is often hard for the corporation administrators to define specific rules for a large number of local resources located at the corporation units. It would be unfeasible to force a policy administrator to write fine grained policy rules applicable to many different subjects in order to maintain the consistency of the corporation policies. Typically, the adopted strategy is to delegate the management of local resources to a local administrator, at the cost of no guarantees of the correct enforcement of corporation's headquarters policies.

Cooperation amongst corporation units in a project requires access to resources located at different company sites. Moreover, a project member may need to work temporarily at different sites of the same company. This requires an access control model that permits both pre-established and dynamic policies configuration at the unit's provider. Appling a bootstrap method for pre-configuring (provisioning) policies at units' providers, it becomes possible to decrease the dependence of the corporation security entities, but a synchronization mechanism is required to maintain the consistence of these policies.

Certificates can be applied to manage access rights, reducing the dependency of a central authorization server and decreasing the number of messages exchanged between the entities of the architecture. Nevertheless, the processing delay for securing an application based on certificates is high.

This paper proposes a unified administration of corporation policies for service oriented environments. Moreover, the management of access rights is based on certificates and pre-configured policies providing autonomy to the corporation's units and decentralizing securely the administration of rights without violating the corporation policies. In other words, a principal that wants to access a corporate unit's local resource, for example, should simply present a certificate to the local guardian of that resource and will obtain access authorization automatically. The granting of local rights must be derived from the set of rights assigned through authorization certificates received from the corporation's headquarters. This scheme avoids violations of

headquarters policies. Additionally, such local granting procedure will update the repository of corporation policies, without requiring any intervention from security administrators.

The remainder of this paper is organized as follows. Section 2 addresses some security issues regarding Web Services. Section 3 briefly introduces provisioning and authorization certificates. Section 4 presents our proposal, while Section 5 describes an example scenario. Section 6 discusses related work. Finally, conclusions are drawn in Section 7.

2. WEB SERVICES

Web Services adopt a set of specifications to provide platform neutral message exchanges and end-to-end security.

WS-Security [1] aggregates extensions to *Simple Object Access Protocol* (SOAP) messages providing support for timestamp, signature and encryption and for representing security credentials (e.g. assertions of SAML, *Security Assertion Markup Language*).

The *Security Token Service* (STS) assures the validity of a security credential or promotes the translation between them.

WS-Trust [2] extends the WS-Security using a request-reply message model for the transportation of credentials in a secure way. Credentials are obtained from the STS and are used to achieve intra and inter-domains trust relationships.

WS-Policy [3] and *WS-SecurityPolicy* [4] define the security context and the minimal requirements for a secure interaction between Web Services.

The *XML Key Management Service* (XKMS) is a service to assist the clients to store and retrieve their keys [5]. The XKMS conceals from the programmer the particular details in the management of each infrastructure (e.g. SPKI/SDSI) by the adoption a neutral scheme to operate with store/retrieve of keys.

The *Service Provisioning Markup Language* (SPML) defines a set of standard operations for the pre-configuration of distributed objects [6], i.e. the Web Services are preset with policies that will regulate the client actions during the services access.

The SAML specifies the transportation of information in a standard format trusted by other Web Services, considering there are pre-established relationships. SAML [7] defines three types of expressions (authentication, attributes, and authorization) which are created by a trusted third party (e.g., STS) and inserted into an assertion that will be associated to a principal.

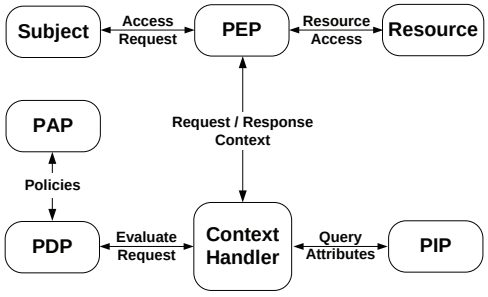

Figure 1. XACML Architecture [8]

The *Extensible Access Control Markup Language* (XACML) defines a XML-based language to write access control policies and a server-based architecture for their evaluation [8]. XACML applies the *Policy Enforcement Point* (PEP) and the *Policy Decision Point* (PDP) entities, defining a request-response context for a standard communication between them (event *Request/Response Context*, Figure 1). In such architecture, the *Context Handler* is an entity that retrieves additional information (e.g., subject and resource attributes) from the *Policy Information Point* (PIP) whenever needed (event *Query Attributes*). Moreover, the Context Handler intermediates the communication between PDP and PEP in order to maintain the messages that they exchange in a standard format. The PAP (*Policy Administration Point*) stores and retrieves all policies (event *Policies*) for the PDP evaluation in a standard format (event *Evaluate Request*).

3. PROVISIONING MODEL AND AUTHORIZATION CERTIFICATES

Using the provisioning access control model, the PEP sends a message to the PDP at initialization time, informing its features and it gets back the corresponding policies regarding the resources it manages. All retrieved policies from PDP's PAP are stored in a PEP's LPAP (*Local PAP*). In this way, the authorization policy can be evaluated locally by the LPDP (*Local PDP*). According to such evaluation, the access request can be either released or blocked by PEP.

The provisioning model is more autonomous than the traditional security architectures, because the PEP does not depend on an external entity (PDP) to evaluate an access request.

An egalitarian model is adopted both in SDSI (*Simple Distributed Security Infrastructure*) and SPKI (*Simple Public Key Infrastructure*). This means that that any principal (user) may issue authorization certificates. Rights are granted by delegation from a principal to another one, creating a chain of certificates. A principal (subject) of an authorization certificate becomes the principal issuer of the next certificate in the chain. The digital signature used on certificates ensures the authenticity of rights delegation and avoids authorization forgery.

An SPKI/SDSI example of authorization certificate coded in S-Expression [9] is shown in Figure 2. Rows 2 to 8 define the certificate issuer (rights' grantor), whilst rows 6 to 8 identify the public key that represents the issuer. Rows 9 and 10 contain the subject of the certificate (rights' grantee). Row 11 indicates that the certificate can be delegated. Row 12 contains the authorization granted by the certificate. In this case, the subject has *writing* rights for all files in the directory *developer* at *www.corporation.com*. In Figure 2, only the main pieces of the authorization certificate are shown.

```
1. (cert
2.  (issuer
3.   (public-key
4.    (rsa-pkcs1-md5
5.     (e #11#)
6.      (n |ALNdAXftavTBG2zHV7BEV59gntNlxtJYqfWIi2kTcFIgIPSjKlHleyi9s
7.      5dDcQbVNMzjRjF+z8TrICEn9Msy0vXB00WYRtw/7aH2WAZx+x8erOWR+yn
8.      1CTRLS/68IWB6Wc1x8hiPycMbiICAbSYjHC/ghq2mwCZO7VQXJENzYr45|))))
9.    (subject
10.    (object-hash (hash md5 |vN6ySKWE9K6T6cP9U5wntA==|)))
11.  (propagate)
12.  (tag (http://www.corporation.com/developer (* set write))))
```

Figure 2. SPKI/SDSI Authorization Certificate

4. THE ARCHITECTURE

The main goal of this work is to provide the decentralization of policy administration and management of access rights, maintaining the unified control in the corporation's headquarters. Let us assume the corporation is composed of different units located in distinct countries. A unit can be a provider or a client, and a user can be client of several providers belonging to the same corporation.

The decentralization of policiy management is achieved through the delegation of rights through authorization certificates. Administrators of local providers can only derive rights from the chain of authorization certificates delegated by the corporation's administrator. The rights granted by the local administrator are used to automatically create new policies and to update the repository of policies in the corporation's headquarters. The provisioning (preset) of policies is used to store them in the repositories of each provider.

Figure 3 shows an overview of the proposed policy control architecture. The Corporation PAP_{Co} (Policy Administration Point) maintains all the policies for all system resources, i.e. the PAP_{Co} is the only policy repository that is subject to administrative updates (*Management* event). The $LPAP_{Pr}$ (Local PAP_{Pr}) stores a local copy of all policies specifically applied to the resources that Provider´s PEP_{Pr} controls (*Provisioning* event).

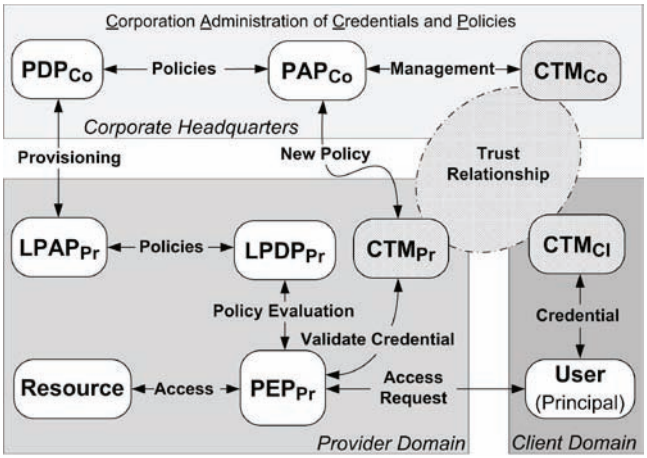

Figure 3. Overview of proposed architecture

The public key infrastructure, which is independent of the underlying technology, supports the delegation of rights via a chain of authorization certificates (*Credential* event). This chain allows keeping track of who delegates rights to whom. Moreover, the corporation administrator specifies which rights can be delegated to others. Such procedure prevents violations in rights granting in the provider's domain, which may compromise the corporation policy rules. Chains of authorization certificates grant rights to users (principals) without the need of administrators' intervention. Also, a properly authorized principal may forward rights to other principals.

When a principal sends a chain of authorization certificates together with an access request (*Access Request* event) to a guardian (Provider's PEP_{Pr}), the latter forwards it (*Validate Credential* event) to a *Credentials and Tokens Manager* (CTM_{Pr}). The CTM_{Pr} is responsible for reducing (converting) the chain to a single certificate.

Based on the reduced (single) certificate, a specific authorization credential is created and sent to the mechanism implementing the PEP_{Pr}. Additionally, such reduced certificate automatically updates the PAP_{Co} (*New Policy* event) as an administrative action that produces specific policies to each resource and user (principal). The $LPAP_{Pr}$ is updated on demand (*Provisioning* event), in order to keep its consistency with the corporation policies stored in PAP_{Co}.If for some reason the $LPAP_{Pr}$ cannot be updated, a pendency is generated via a notification sent to the *Corporation Administration of Credentials and Policies* (CACP). Such scheme favors decentralization of policy administration and, at the same time, favors the synchronization of repositories.

By using certificates, it is possible to create mutual trust relationships between the CTMs. The *Trust Relationship* (Figure 3) is based on mutually issued CTMs credentials. Such trust relationships were used for administrative purposes, mainly to allow the granting of rights between administrators (e.g. CTM_{Co} to CTM_{Cl}). In other words, for the administrator of CTM_{Co} to be able to grant rights to CMT_{Cl}, it is necessary to establish a mutual *Trust Relationship*. Such a procedure is the only way a principal can securely know the public key of another principal, given that a serverless public key infrastructure has been used. Trust relationships are considered administrative needs, because the administrators of CTM do not use delegated rights. Normally, such rights are partially forwarded (delegated) to users within a Client Domain (*Credential* event).

The transposition of provider domains is easily obtained using authorization certificates, i.e., clients from diverse corporation's domains (sites) can obtain access to corporation providers without the need of an account either on the headquarters or on the provider side. The authorization certificates carry all the attributes needed to evaluate an access request. The service provider is not required to contact authorities in the client domain to obtain additional attributes, thus the provider can take an authorization decision based only in the chain of certificates.

5. APPLICATION SCENARIO

The administration of access control policies is not an easy task, especially within an environment having employees of several different ranks and many departments distributed across several corporation units. Additionally, access control systems are heterogeneous, since they must fulfill unit-specific requirements.

At the bootstrap phase, the provisioning is carried out employing SPML. The Provider PEP_{Pr} sends a message to PDP_{Co} (event 1, Figure 4) asking for the configuration of its policy repository. The PDP_{Co}, in its turn, retrieves the corresponding policies stored in the PAP_{Co} (event 2) and send it to the $LPAP_{Pr}$ (event 3).

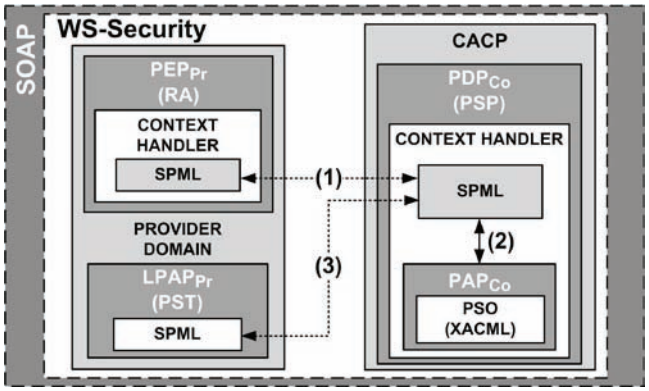

Figure 4. Policy Provisioning

According to the SPML (Figure 4), the proposed architecture (Figure 3) should be viewed as described below. The PDP_{Co} represents the PSP (*Provisioning Service Provider*), while the PAP_{Co} – the repository of XACML policies – acts as a PSO (*Provisioning Service Object*). The PEP_{Pr} represents the RA (*Requesting Authority*), and the $LPAP_{Pr}$ represents the PST (*Provisioning Service Target*). The PST stores all the provisioned policies. The provisioning of all policies is only executed at the

PEP_{Pr} bootstrap. PDP_{Co} can administer many PEP_{Pr} (each one in a specific corporation unit, for instance) and each PEP_{Pr} can manage many different resources.

Trust Relationships established between the Credentials and Tokens Manager (CTMs, Figure 3) are based on SPKI/SDSI mutual CTM group inclusion. In other words, each CTM inserts the other CTM in its local SPKI/SDSI group and issues a certificate denoting group membership (a SDSI name certificate). The group membership serves as the basis to ensure that an entity which triggers a message can be trusted.

In a typical scenario, a principal issue an access request for a given resource at any given time. The *Context Handler* attached to PEP_{Pr}, is responsible for forwarding this request to the $LPDP_{Pr}$ which queries $LPAP_{Pr}$. If the $LPDP_{Pr}$ does not find any policy that applies to that resource, then it is not possible evaluated the access requests locally. That is, to get access to a resource, a principal should be inserted in the policy rules of the $LPAP_{Pr}$. In this case, the $LPDP_{Pr}$ notifies the PEP_{Pr} that it is not able to evaluate the request.

The PEP_{Pr}, in its turn, offers an alternative to the user, i.e. the user may obtain the required rights through a chain of authorization certificates. Thus, employing the challenge-response protocol [10], the PEP_{Pr} sends back to the user (principal) a message containing a WS-Policy document informing which rights are required to get access to the resource.

A user on the client's domain gets authentication from the CTM_{Cl} through the digital signature of a request containing the target resource (event 1 in Figure 5). Assuming that the CTM_{Cl} is the administrator of a department, for instance, then the rights coded in the authorization certificate can be delegated. Thus, if the required rights to access the resource are locally available on a chain, the CTM_{Cl} delegates it to user. If not, the CTM_{Cl} acquires the rights for the user through the *Trust Relationship* (event 2).

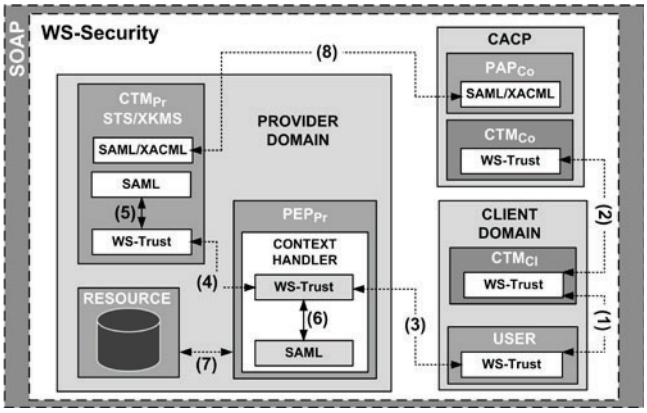

Figure 5. Getting SPKI/SDSI Credential

When the authorization chain is finally obtained, the CTM_{Cl} sends it back to the user, which sends both the request and the certificates to the PEP_{Pr} (event 3). The PEP_{Pr} forwards the chain of certificates to the CTM_{Pr} (event 4) in order to obtain the SAML assertion. After that, the CTM_{Pr} sends the chain to the XKMS, which reduces the SPKI/SDSI certificate chain and sends back a single certificate to CTM_{Pr}.

The CTM_{Pr} generates an SAML assertive (native credential in Web Services) based on the rights extracted from the single certificate and marshals it in a WS-Trust reply message (event 5, Figure 5). The CTM_{Pr} sends back a SAML assertion to the PEP_{Pr}, which releases the access to the resource (event 7, Figure 5) according to the expressions holding in the unmarshalled assertion (event 6). The SAML assertion contains the authorization and attributes expressions.

In addition to providing the SAML assertion, even though it is still based on the reduced certificate, the CTM_{Pr} generates an XACML policy coded in an SAML message that will be sent to the PAP_{Co} for update purposes (event 8). This procedure is done in accordance with the SAML 2.0 profile of XACML v2.0 [11].

The PAP_{Co} stores a local copy of the received policy and sends a policy update message addressed to the $LPAP_{Pr}$ through the SPML infrastructure (Figure 4). This automatic policy creation procedure is equivalent to a policy insertion action performed by a human administrator.

5.1 Prototype Implementation

The prototype was implemented in Java integrating the following projects: OpenSPML (www.openspml.org), OpenSAML and SAML 2.0 Profile for XACML v2 (www.opensaml.org), and SUN XACML (sunxacml.sourceforge.net). The TomCat server (tomcat.apche.org), the SOAP engine Axis2 (ws.apache.org/axis2), the Rampart module for WS-Security and WS-Trust support, and the Apache Neethi for WS-Policy support (ws.apache.org/commons/neethi) from Apache Software Foundation were equally used.

The repositories of policies used Oracle Berkeley Data Base XML (www.oracle.com), and the SPKI/SDSI APIs implemented by Morcos [12].

Figure 6 shows the main tags of an XACML policy generated from a reduced chain of SPKI/SDSI certificates. Row 5 shows the policy subject (principal SPKI/SDSI) and row 10 indicates the target resource. Considering that the certificate chain is valid, row 14 uses default value "*Permit*", otherwise the policy would not be generated. Row 17 defines the authorized operation. Since there is only one XACML policy generated from each reduced certificate chain, we adopted the "*first-applicable*" XACML rule combination algorithm (row 1).

```
1. <Policy PolicyId="WritePolicy" RuleCombiningAlgId="first-applicable">
2.   <Target>
3.     <Subjects> <Subject> <SubjectMatch MatchId="function:string-equal">
4.       <AttributeValue DataType="string">
5.         vN6ySKWE9K6T6cP9U5wntA==
6.       </AttributeValue>
7.     </SubjectMatch> </Subject> </Subjects>
8.   <Resources> <Resource> <ResourceMatch MatchId="function:anyURI-equal">
9.     <AttributeValue DataType="anyURI">
10.      http://www.corporation.com/developer
11.    </AttributeValue>
12.  </ResourceMatch> </Resource> </Resources>
13. </Target>
14.   <Rule RuleId="WriteRule" Effect="Permit">
15.     <Target> <Actions> <Action> <ActionMatch MatchId="function:string-equal">
16.       <AttributeValue DataType="string">
17.         write
18.       </AttributeValue>
19.     </ActionMatch> </Action> </Actions> </Target>
20.   </Rule>
21. </Policy>
```

Figure 6. XACML policy based on Figure 2

SPKI/SDSI authorization certificates carrying more than one access right for the same principal and referencing the same resource can generate XACML policy rules with more than one *<Action>* tag (rows 15 to 19). In Figure 6, the policy validity condition supplied by SPKI/SDSI certificate and some other none essential XML tags were omitted to facilitate the understanding.

It is important to notice that the scenario explored in this paper can be extended to more complex ones.

5.2 Prototype Evaluation

The evaluation was done over the scenario presented in Section 5. Our goal was to observe time consumption in the main entities of the proposed architecture against the total time of a request.

On the local evaluation (provisioning model), the PEP_{Pr} spent 13% of the RTC (*Response Time for the Client*) dealing with XML messages and enforcing the decisions received from the $LPDP_{Pr}$, which spent 2% of the RTC.

On the evaluation using only SPKI/SDSI certificates, the PEP_{Pr} spent 20% of the RTC for marshalling/unmarshalling messages (chain with 2 certificates) and enforcing the resulting SAML assertion. The CTM_{Pr} spent an amount of 60% of the RTC – 40% of which for validating the chain of certificates and creating XACML policies, and 5% for generating SAML assertion. The remainder time (15%) is spent on internal marshalling/unmarshalling of messages on the CTM_{Pr}.

The usage o SPKI/SDSI has shown that with the use of authorization certificates is computationally more costly than provisioning. However, one should notice that SPKI/SDSI is only applied to create a policy that does not exist, without human interference. From that point on, the provisioning is always applied in the subsequent evaluation of the same access request.

6. Related Work

The architecture presented in [13] proposes the use of a trusted intermediary that should be responsible for dealing with distinct security technologies. The mediator retrieves attributes from the client domain and exchanges SPKI/SDSI and X.509 credentials for SAML assertions to obtain interoperability. However, the proposal induces a tight dependency among the architectural entities, which is contrary to the desired loose coupling of Web Services.

The work in [14] proposes an access control system based on attributes which can cross security domains, enabling easy interactions of heterogeneous systems. The proposed model provides access to Web Services based on a signed digital credential, and additional attributes provided by trusted authorities. The proposed approach is based on the traditional WS security. Thus, this proposal suffers of the same strong coupling problem of the previous work.

7. Conclusion

This paper presented a proposal to maintain the unified control of access rights and access control policies to be applied within a corporate environment, using a scheme for distributed policy writing, evaluation and enforcement. In fact, policies are written on demand in each provider domain, without the intervention of the administrator, and they are automatically and securely updated on the PAP_{CO}.

The proposed approach is compliant with the server-based model of Web Services. Nevertheless, it does not depend on an infrastructure based on authentication and authorization servers to cross security domains, thus favoring loose coupling among the architectural entities.

The architecture defined requires less message exchanging than traditional security commonly applied to Web services, being more appropriate to deal with the local autonomy of the provider.

The architecture tolerates a possible unavailability of the CACP, in the corporation's headquarters, since a local copy of the policies applicable to each provider is preset during the bootstrap phase. Moreover, in the case the local policies do not contain rules for a certain principal, the SPKI/SDSI certificate chain can grant rights to principals even if the CACP is inactive. This approach is distinct from the traditional ones that, in such cases, would simply deny the request.

The prototype implementation based on SPKI/SDSI and provisioning, deals with the requirement of loose coupling of Web Services. Besides, it gives operational autonomy and reduced the dependence on messages exchanges between the architectural entities. Additionally, this combination shows that it is possible to reduce the human and computational costs involved in policy management while improving Web Services security.

8. REFERENCES

[1] OASIS. Web Services Security: SOAP Message Security 1.1 - WS-Security v 1.1. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#wssv1.1.

[2] OASIS. WS-Trust 1.3 Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#wstrustv1.3.

[3] W3C. Web Services Policy Access: April 2008. Available at: http://www.w3.org/TR/ws-policy/.

[4] OASIS. WS-SecurityPolicy v 1.2. Access: Jan. 2008. Available at: http://www.oasis-open.org/specs/index.php#wssecpolv1.2.

[5] W3C. XML Key Management Specification - XKMS v 2.0. Access: Sep. 2007. Available at: http://www.w3.org/TR/xkms2/.

[6] OASIS. Service Provisioning Markup Language - SPML v 2. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#spmlv2.0.

[7] OASIS. Assertions and Protocols for the OASIS Security Assertion Markup Language - SAML v 2.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#samlv2.0.

[8] OASIS. eXtensible Access Control Markup Language - XACML v 2.0. Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#xacmlv2.0.

[9] C. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, 1999.

[10] NIST. Entity Authentication Using Public Key Cryptography. FIPS PUB 196. Access: Sep. 2007. Available at: http://csrc.nist.gov/publications/fips/fips196/fips196.pdf.

[11] OASIS. SAML 2.0 profile of XACML v2.0 Access: Sep. 2007. Available at: http://www.oasis-open.org/specs/index.php#samlv2.0.

[12] A. Morcos, "A Java Implementation of Simple Distributed Security Infrastructure," in *EECS*. Master Dissertation. Massachusetts Institute of Technology, 1998.

[13] E. R. Mello and J. S. Fraga, "Mediation of Trust across Web Services," in *ICWS'05*. IEEE, 2005.

[14] S. Hai-bo and H. Fan, "An Attribute-Based Access Control Model for Web Services," in proceedings of PDCAT'06, IEEE, 2006.