SECURITY FOR WEB SERVICES - STANDARDS AND RESEARCH ISSUES

by L. D. Martino, E. Bertino

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Security for Web Services - Standards and Research Issues

Lorenzo Martino (1) and Elisa Bertino (2)

(1) Department of Computer Technology and Cyber Center
Purdue University, West Lafayette, IN
lmartino@purdue.edu
(2) Department of Computer Science and CERIAS
Purdue University, West Lafayette IN
bertino@cs.purdue.edu

Abstract. This chapter identifies the main security requirements for Web services and it describes how such security requirements are addressed by standards for Web services security recently developed or under development by various standardizations bodies. Standards are reviewed according to a conceptual framework that groups them by the main functionalities they provide. Standards that are covered include most of the standards encompassed by the WSS roadmap [2]; the Security Assertion Markup Language -SAML-, WS-Policy, XACML, that is related to access control and has been recently extended with a profile for Web services access control; XKMS and WS-Trust; WS-Federation, LibertyAlliance and Shibboleth, that address the important problem of identity management in federated organizations. Finally, issues related to the use of the standards are discussed and open research issues in the area of access control for Web services and innovative digital identity management techniques are outlined.

1 Introduction

Today Web services are a fundamental component of agile e-business. Through the use of eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), and related open standards deployed in Service Oriented Architectures (SOA), they allow data and applications to interact through dynamic and ad hoc connections. Web services technology can be implemented in a wide variety of architectures, can co-exist with other technologies and software design approaches, and can be adopted in an evolutionary manner without requiring major transformations to legacy applications and databases. Despite the heterogeneity of the underlying platforms, Web services enhance interoperability and are thus able to support business applications composed by chains of Web services. Interoperability is a key promise of Web service technology and therefore notions such as Web service composition and technologies like workflow systems are being investigated and developed.

The use of Web services, stand-alone or composed, must however provide strong security guarantees. Security is today a relevant requirement for any distributed application, and in particular for these enabled by the Web such as e-health, e-commerce, e-learning. Providing security guarantees in open dynamic environments characterized by heterogeneous platforms is however a major challenge. Web services security encompasses several requirements that can be described along the well known security dimensions:

– *integrity*, whereby information can be modified only by users who have the right to do so, and only in authorized ways. In particular, message integrity requires that a message remain unaltered during transmission. Ensuring information integrity might also require that information is transferred only among intended users and in intended ways.
– *confidentiality*, whereby information can be disclosed only to users authorized to access it. When applied to messages, it requires that the content of a message cannot be viewed while in transit, except by authorized services.
– *availability*, whereby the use of the system cannot be denied to entitled users inadvertently or due to denial of service attacks by a malicious party.
– *accountability*, whereby users are accountable for their security-relevant actions. A particular case of this is non-repudiation, where responsibility for an action cannot be denied.

Moreover, each Web service must protect its own resources against unauthorized access. This in turn requires suitable means for: *identification*, whereby the recipient of a message must be able to identify the sender; *authentication*, whereby the recipient of a message needs to verify the claimed identity of the sender; *authorization*, whereby the recipient of a message applies access control policies to determine whether the sender has the right to use the required Web services and the protected resources.

In a Web service environment it is however not enough to protect the service providers, it is also important to protect the parties requiring services. Because a key component of the Web service architectures is represented by the discovery of services, it is crucial to ensure that all information used by parties to this purpose be authentic and correct. Also we need approaches by which a service provider can prove its identity to the party requiring the service in order to avoid attacks, such as *phishing attacks*.

Within this context, the goal of securing Web services can be decomposed in three broad subsidiary goals:

– Providing mechanisms and tools for securing the integrity and confidentiality of messages as well as the guarantee of message delivery.
– Ensuring that the service acts only on message requests that comply with the policies associated with the services.
– Ensuring that all information required by a party in order to discover and use services is correct and authentic.

Different security mechanisms and tools have been developed and deployed over time to this end. The overall goal of Web services security standards is to make interoperable different security infrastructures and to reduce the cost of security management. To achieve this goal, Web services security standards have to provide a common framework, and common protocols, for the exchange of security information between/among Web services that, once implemented and deployed:

– can accomodate such existing heterogenoeus mechanisms, i.e. different encryption algorithms, different access control mechanisms, etc.
– can be extended so as to cope with new requirements and/or available security technologies.

This chapter surveys some existing and proposed standards for Web services security and some recent research proposals for access control for Web services. The chapter is organized as follows. Section 2 introduces the different notions of standards. This classification gives indications about the maturity, the stability and the level of endorsement of a standard. Then a framework for the security standards is presented. In this framework, Web services security standards are conceptually grouped depending on the various aspects of Web services security they address. Then, for each security aspect of the framework, the related standards are briefly surveyed, describing their specific purpose, their main features and their current status. The section concludes with a brief discussion on the main issues concerning the adoption of such standards. Section 3 outlines recent research proposals and discusses open research issues. Section 4 outlines some conclusions.

2 Web services security standards framework

In this section we present first the different notions of standards. We then present the conceptual framework for Web services security standards, and, for each level of this framework, we survey existing and proposed standards, their specific purpose, and their current status.

2.1 The concept of standard

The concept of “standard” covers different notions, ranging from a public specification issued by a set of companies, to a de jure standard issued by a recognized standardisation body. These different notions can provide to the potential users useful indications about the maturity, the stability and the level of endorsement of a given standard. The following “types” of standards can be distinguished:

1. *De facto standards:* a technology that is used by a vast majority of the users of a function. Such function may for example be provided in a product from a single supplier that dominates the market; or it may be a patented technology that is used in a range of products under license. A *de facto* standard may be endorsed by a standardisation initiative, and eventually become a consortium recommendation, or a *de jure* standard. The relevant requirements are that it is widely used, meets the needs for functionality, and supports interoperability.

2. *De jure standards:* standards defined by entities with a legal status in international or national law such the International Organization for Standardization -ISO-, national standards bodies (e.g. the BSI British Standards in the UK, the American National Standards Institute -ANSI- in the US) or continental standards (e.g. European standards). These standards are relevant in the health and safety related areas, in business quality measures and in long term IT areas. The issuance of a standard by one of these standardization bodies is usually a long lasting process, which can take many years, and requires the agreement by the appropriate committee of the satndardization body.
3. *Consortium recommendations:* a technology agreed on and recommended by a groups of companies in order to fulfil some functionality. Such consortia may vary in size from groups of a few large manufacturers (e.g. Microsoft, IBM and BEA) to much larger groups or organizations such as the Organization for the Advancement of Structured Information Standards (OASIS), the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

De facto standards, eventually promoted to the jure standard by a subsequent endorsement by a standardization body, offer a higher guarantee of support for interoperability. Conversely, de jure standards or consortia recommendations do not guarantee per se that a standard will be widely endorsed nor the market availability of really interoperable implementations by multiple vendors. Moreover, the definition of a standard and its issuance by a standardization body or by a consortium is a long lasting process, subject to formalized organizational procedures; for example, W3C takes 6 months to establish a working group on a technology, and then 18 months to 3 years to agree a recommendation, which is only released if there are working interoperable implementations of all functions in the technology, and enough of the members of W3C support it. The development stage of a standard within a standardization body can give an indication of the maturity level of the standard itself.

2.2 The framework for Web services security standards

Web services security standards address a variety of aspects, ranging from the message level security to the identity management. In order to provide a structured and engineered approach to the development of the standards, an overall conceptual reference framework was needed. Such a reference framework is crucial in organizing the standards according to layers and in promoting the reuse of already developed specification. The first structured framework was proposed in April 2002, by Microsoft and IBM in the white paper: “Security in a Web Services World: A Proposed Architecture and Roadmap” [2] describing a strategy for addressing security requirements in a web services environment. As integral part of the strategy, the development of a set of composable standard specifications was foreseen. As shown in Figure 1, the Web Services Security (WSS)

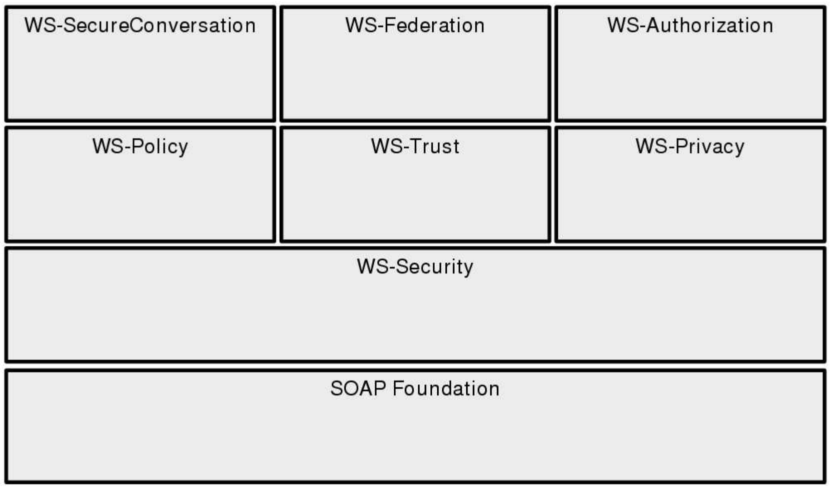

Fig. 1. WSS Security standard framework

specifications encompass different specifications, each addressing specific aspects of security.

According to the roadmap [2], WS-Security was to provide a message security model and the specification of mechanisms to attach signature and encryption headers to SOAP messages. WS-Policy was intended to describe: (1) the security policies, such as required security tokens and supported encryption algorithms, as well as more general policies adopted by a Web service; (2) the mechanisms by which trusted SOAP message exchanges could be built. WS-Trust was intended to define the model for establishing both direct and brokered trust relationships (including third parties and intermediaries) through the creation of security token issuance services. WS-Privacy would have to define a model for embedding a privacy language into WS-Policy and for associating privacy claims with a message in WS-Security.

On top of such standards, further follow-on specifications were envisaged. WS-SecureConversation was introduced with the goal of extending the single message security provided by WS-Security to a conversation consisting of multiple message exchanges, whereas WS-Federation was introduced with the goal of describing how to manage and broker trust relationships in a heterogeneous federated environment. Finally, the goal of WS-Authorization was to provide a support for the specification of authorization policies and for managing authorization data.

It is worth noting that the specifications for WS-Authorization and WS-Privacy followed a different development with respect to the other standards of the roadmap. In particular, WS-Authorization was replaced by the specification of XACML (see section 2.3), whereas WS-Privacy does not seem to have received the same level of effort, but rather it was addressed by manufacturer proposals such as the IBM Enterprise Privacy Authorization Language (EPAL) [32].

With respect to the original framework, we adopt a slightly different classification, as shown by Figure 2. This classification has been adopted in order to take into account in the discussion the standards below the SOAP layer and, most importantly, to group the standards by their main intended purpose rather than adopting a "stack" view that emphasizes mainly how each specification is built on top of the other ones. In particular, we deemed useful to separate message-level security specifications (the two groups labelled Message Security and Reliable Messaging) from the specifications addressing Policy and Access Control, Security Management, and Identity Management issues.

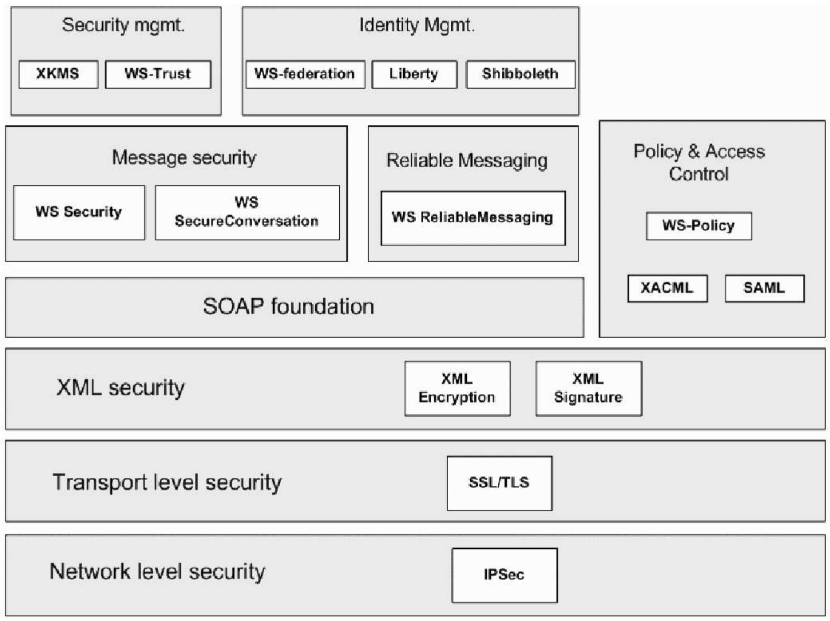

Fig. 2. Refined classification of standards

2.3 An Overview of Current Standards

"Near the wire" security standards At the transport layer, the well known Secure Socket Layer [3] and the Transport Layer Security [4], collectively referred as SSL/TLS, are the de facto standards used to assure transport level security for Web services applications. SSL/TLS provides a fast, efficient and widely accepted protocol for mutual authentication of end systems, data message authentication and optional confidentiality. SSL/TLS enables point-to-point secure sessions. Encrypting the communication between a browser and a Web service is a fairly safe procedure because the connection is exclusive to the client browser and the Web server that acts as the gateway to internally hosted application logic. However, a message transmitted by a client, such as browser or an application, might be routed (and processed) by a number of intermediary services before reaching its destination. SSL protects the message contents only while being transmitted between these intermediaries but it cannot protect the message when it is processed by an intermediary service. At the network layer, IPSec is a de jure standard for transport security that may become relevant for Web services. IPSec provides security services including access control, connectionless integrity, data origin authentication, protection against replays (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality.

XML-encoded message security Because XML is used as the language to encode messages exchanged among Web services, securing XML data is a key requirement. Several standards have thus been developed to support encryption and digital signatures for XML data as well as authentication protocols.

XML Encryption and XML Signature. The messages exchanged between a client and a Web service are encoded as XML "documents". These two key standards specify how to protect the actual content within XML documents. The XML-Encryption specification [6] contains a standard model for encrypting both binary and textual data, as well as a means of communicating information essential for recipients to decrypt the contents of received messages. XML Encryption is a W3C Recommendation and it is the commonly accepted standard for encryption. XML-Digital Signatures [7] defines a standardized format for representing digital signature data. Digital signatures assure the recipient of the message that the message was in fact transmitted by the expected sender. It also provides a means of communicating the message contents so that they cannot altered while in transit, as well as support for standard non-repudiation so that the sender of the message can not deny of having sent it. Like the XML-Encryption standard, XML-Digital Signature also supports binary and textual data. XML-Digital Signature is a W3C Recommendation and it is the commonly accepted standard for digital signatures.

WS-Security. SOAP Message Security ("WS-Security") [8] has been specified by OASIS and is the commonly accepted standard for message security in Web and Grid Services. WS-Security describes enhancements to SOAP messaging to provide single-message origin authentication and single-message confidentiality. It relies upon XML Encryption and XML Signature standards and achieves its objectives by attaching and including security tokens within SOAP messages through a general-purpose mechanism. A (software) security token is a representation of security-related information (e.g. X.509 certificate, Kerberos tickets and authenticators, mobile device security tokens from SIM cards, username, and so forth). WS-Security specifies a general-purpose mechanism for associating security tokens with messages, without requiring the use of any specific type of security token. Various security token formats have been specified for use with WS-Security, including username/password (OASIS), SAML assertions 2.3

(OASIS), XrML/REL tokens (OASIS), X.509 certificates (IETF and OASIS), Kerberos tickets (OASIS). For security tokens that are not encoded in XML, such as X.509 certificates and Kerberos tickets, WS-Security provides a mechanism for encoding binary security tokens. Due to the variety of supported security token formats, WS-Security is very flexible; moreover, it can be extended with profiles to support new security tokens.

Message integrity is provided by using XML Signature in conjunction with security tokens (which may contain or imply key data). WS-Security supports multiple signatures, potentially by multiple actors, and it can be extended to support additional signature formats. The signatures may reference (i.e. point to) a security token. WS-Security provides message confidentiality by encrypting portions of the SOAP message, using XML Encryption in conjunction with security tokens. The encryption mechanisms are designed to support additional encryption technologies, processes, and operations by multiple actors. The encryption may also reference a security token. WS-Security is a consortium recommendation.

WS-SecureConversations. The interactions between a client and a Web service, or between two Web services typically consist of multiple message exchanges. Thus, it is important to secure not only a single message exchange, but also multiple messages exchanges that are needed to complete a meaningful transaction. WS-SecureConversation [9] defines extensions, based on WS-Security and WS-Trust [20], aimed at providing secure communication across multiple messages. These extensions are based on the establishment and sharing of security context between the communicating parties and on the derivation of keys from the established security contexts. A security context is an abstract concept that refers to an established authentication state and it is represented by a Security Context Token. This specification defines three different ways of establishing a security context among the parties of a secure communication: the context initiator can request a Security Token Service, as defined by WS-Trust, to create a security context token, or a security context token can be created by one of the communication parties and propagated within a message; or a security context token can be created when needed through negotiation/exchanges among the participants. This way potentially more efficient keys or new key information can be exchanged, thereby increasing the overall performance and security of the subsequent exchanges. WS-SecureConversation is a consortium revised public draft release.

WS-ReliableMessaging. Guaranteeing the integrity and the confidentiality of the messages does not avoid that messages be lost, duplicated or reordered. Correct delivery of messages must also be assured. Deliverys guarantee is provided by several middleware components implementing the "store and forward" paradigm, such as Microsoft Message Queuing (MSMQ) [29], IBM Messaging and Queuing (WebSphere, MQ) [30], or Sun Java System Message Queue[31]. WS-ReliableMessaging [10] defines a messaging protocol to identify, track, and manage the reliable delivery of messages between exactly two parties, a source and a destination endpoints (referred to as the Reliable Messaging RM - Source and Reliable Messaging RM - Destination, respectively), despite the presence of software component, system, or network failures. The protocol supports the communicating endpoints in providing delivery assurances. It is the responsibility of the RM Source and RM Destination to fulfil the delivery assurances, or raise an error. Endpoints can provide four basic delivery assurances:

– AtMostOnce assurance guarantees that messages will be delivered at most once without duplication, or that an error will be raised on at least one endpoint. It is possible that some messages in a sequence may not be delivered.
– AtLeastOnce assurance guarantees that every message sent will be delivered, or an error will be raised on at least one endpoint. Some messages may be delivered more than once.
– ExactlyOnce assurance guarantees that every message sent will be delivered without duplication, or an error will be raised on at least one endpoint.
– InOrder assurance guarantees that messages will be delivered in the order that they were sent. This delivery assurance may be combined with any of the above delivery assurances. It does not provide any assurance about message duplications or omissions.

The messaging protocol defined in this specification can be implemented using different network transport technologies. However, this specification defines a SOAP binding in order to support interoperable Web services. WS-ReliableMessaging specification will be submitted to OASIS for further refinement and finalization as a Web services standard.

Access Control standards *Security Assertions Mark-up Language* The Security Assertions Mark-up Language (SAML) [22] is an XML based framework, developed by OASIS, to support the exchange of security information, also called trust assertions, between online business partners, such as vendors, suppliers, customers, over the Internet. The applications and the environments that can use SAML are quite varied, from simple browser-based applications to more complex n-tiered architecture web services. Security information takes the form of security assertions, where an assertion states certain facts (characteristics and attributes) about a subject. The current SAML framework supports three kinds of security assertions: Authentication, Attribute and Authorisation decisions. An Authentication assertion states that the subject S was authenticated by means M at a certain time. It is issued by the party that successfully authenticated the subject. An Attribute assertion states that the subject S is associated with the set of attributes A with values B (for example, that Alice is associated with attribute “Company” with value “Hertz”). An Authorisation decision assertion states which actions the subject S is entitled to execute on resource R (for example, that a user has been authorized to use a given service).

An example of SAML authentication assertion, stating Alice was originally authenticated using a password mechanism at 2006-04-02T19:05:17 is shown in Figure 3.

Assertions are issued by SAML authorities, namely authentication authorities, attribute authorities or policy decision points. While SAML can be used

```
<saml:Assertion
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    MajorVersion="1" MinorVersion="1"
    AssertionID="biuEZCGxcGiF4gIkL5PNltwU7duY1az"
    Issuer="www.it-authority.org"
    IssueInstant="2006-04-02T19:05:37">
    <saml:Conditions
       NotBefore="2006-04-02T19:00:37" NotOnOrAfter="2006-04-02T19:10:37"/>
    <saml:AuthenticationStatement
      AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
      AuthenticationInstant="2006-04-02T19:05:17">
     <saml:Subject>
      <saml:NameIdentifier
        NameQualifier= www.it-authority.org
            Format="http://www.customformat.com/">
            uid=alice
      </saml:NameIdentifier>
       <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
            urn:oasis:names:tc:SAML:1.0:cm:artifact-01
            </saml:ConfirmationMethod>
         </saml:SubjectConfirmation>
    </saml:Subject>
   </saml:AuthenticationStatement>
</saml:Assertion>
```

Fig. 3. An example of SAML assertion

to make assertions about credentials, however it does not provide mechanisms to check or revoke credentials. This means that the party accepting a SAML assertion as true is trusting the SAML authority that issued the assertion.

In order to exchange security assertions between involved parties, SAML defines a request and response protocol that consists of XML-based messages; a client uses this protocol to request a specific assertion or to make authentication, attribute, and authorization decisions queries to a SAML authority and obtain a response from them. These messages can be bound to many different underlying communication and transport protocols. SAML defines also several profiles. Generally, a profile of SAML defines constraints and/or extensions of the core protocols and assertions in support of the usage of SAML for a particular application, by specifying how particular statements are communicated using appropriate protocol messages over specified bindings. For example, the Web Browser SSO Profile specifies how SAML authentication assertions are communicated using the Authentication Query and Response messages over a number of different bindings in order to enable Single Sign-On for a browser user. SAML assumes that the two or more endpoints of a SAML transaction are uncompromised, but the attacker has complete control over the communications channel. Moreover, SAML does not cope directly with two security issues:

– Initial Authentication: authentication assertions convey information about an already happened authentication act. Consequently, such an assertion can be trusted by the party receiving the assertions as far as it trustes the party/authority that made the assertion.
– Trust Model: the security of a SAML conversation will depend on the underlying key management infrastructure (shared key or asymetric key) and hence it is secure as long as the keys used can be trusted. Undetected compromised keys or revoked certificates, for example, could allow a breach of security.

It is worth noting that, as described in the SAML specification itself, the SAML protocol is susceptible to a denial of service (DOS) attack. Moreover, handling a SAML request is potentially a very expensive operation, since it includes parsing the request message (typically involving construction of a DOM tree), database/assertion store lookup (potentially on an unindexed key), the construction of a response message, and potentially one or more digital signature operations. SAML Version 2.0 is a Committee Draft specifications approved for public review by the OASIS Security Services Technical Committee.

eXtensible Access Control Mark-up Language - XACML- . XACML [11] provides an extensible, XML-encoded language for managing authorization decisions. To this end, XACML language allows one to express access control policies and access requests/responses. XACML was conceived as one component of a distributed and inter-operable authorization framework, with the following underlying rationales:

– first, access control policies do not have to be embedded or tightly linked to the system they govern.

– second, XACML policies can be applied to different heterogeneuos resources such as XML documents, relational databases, application servers, web services, etc.
– third, a standard policy exchange format allows different web services to exchange or share authorization policies, as well as to deploy the same policy to heterogeneous systems.

It is worth noting that XACML includes also a non-normative data flow model ([11], Section 3.1 Data flow model), reproduced in Figure 4, that describes the major actors involved in the processing of an access request. This model, that can be considered as an evolution of the ISO 10181-3 model [13], can be used as a reference model for the implementation of a XACML engine.

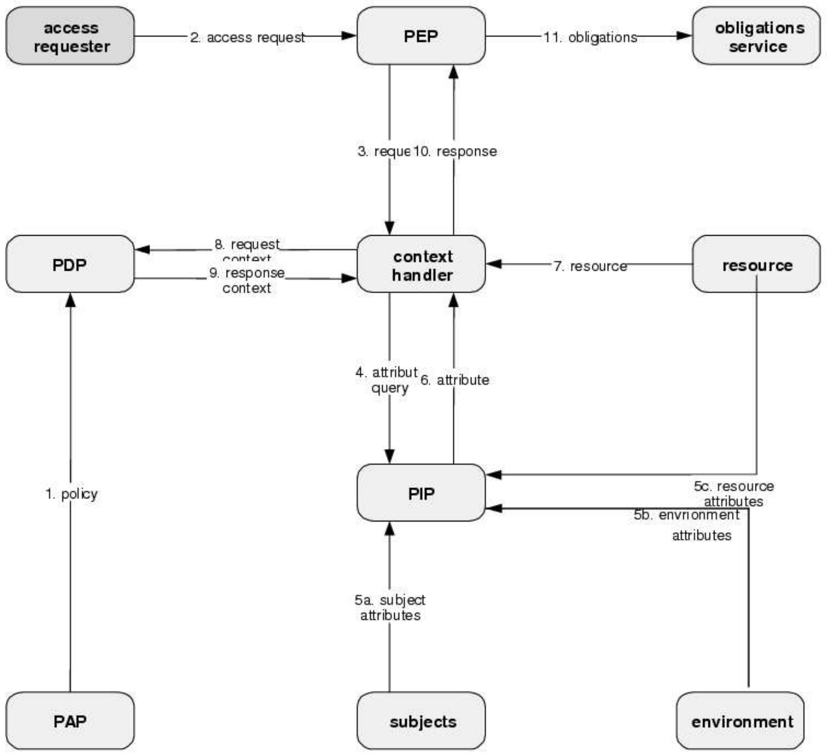

Fig. 4. XACML Data Flow Model

The components of the model above are the following:

– The Policy Administration Point (PAP) creates security policies and stores these policies in the appropriate repository.
– The Policy Enforcement Point (PEP) performs access control by making decision requests and enforcing authorization decisions.

– The Policy Information Point (PIP) serves as the source of attribute values, or the data required for policy evaluation.
– The Policy Decision Point (PDP) evaluates the applicable policy and renders a response to the PEP containing the authorization decision. The possible response values are: Permit, Deny, Indeterminate (in case an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request cannot be answered by this service).

The PEP and PDP might both be co-located within the same application, or might be distributed across different servers.

In XACML, the access request is represented by the Request schema that specificies the requesting Subject, the requested Object and the specific Action requested on the Object. The XACML policy language was designed to be general enough so as to describe general access control requirements. It is also extensible, by means of standard extension points, in order to accomodate the definition of new functions, data types, combining logic, and so forth. In XACML, a policy is the smallest element that the PDP can evaluate. A policy represents a single access control policy, expressed through a set of Rules. A Rule specifies the Target which it applies to and the effect of the rule, that is Permit or Deny. The Target basically models the access request, by means of a set of simplified conditions for the Subject, Resource and Action that must be met, i.e. evaluate to true, for the Rule to apply to a given request. Any number of Rule elements may be used, each of which generates a true or false outcome. Combining these outcomes yields a single decision for the Policy, which may be "Permit", "Deny", "Indeterminate", or a "NotApplicable" decision.

As a a policy example, consider the following one: "MPEG movie for adults cannot be accessed by users with age less than 18 years". The movie is the resource the access to which must be controlled; it will be modelled by an element having an attribute "'category". Similarly, the subject will have an attribute "'age"'. In this case, the policy is composed by a single rule, that specifies the condition: "age less than 18"' for the subject, the condition: category = "'adult only"' for the resource, the condition: "'download"' for the action, and the effect "'Deny"'. More than one Rule (or more than one Policy) may result applicable to a given access request; XACML defines a set of Combining Algorithms to reconcile multiple outcomes into a single decision, namely, Deny-overrides, Permit-overrides, First-applicable, Only-one-applicable.

In XACML a policy can have scopes of different granularity; actually, a policy can apply to all the operations provided by a Web service, or rather to a specific operation of the Web service. The same policy can be "reused" by different services by applying it to different ports. OASIS XACML Profile for Web-Services [12], hereafter referred to as XACML2, is a proposal for extending XACML to deal with the specific characteristics of Web services. There are two main extensions. The first one is the possibility of defining in a precise way the various aspects to which a security policy applies to, for example distinguishing the security policy that must be applied to the message level from the access control policy applied to a Web service or to an operation of the Web service.

The second one is the use of the policy combination mechanisms defined in XACML in order to combine the preference/requirements policy of the Web service client with the access control policy of the Web service provider. XACML2 thus supports the specification of policies for the whole service (a WSDL port), for a Web service operation (a single WSDL operation), for a request message (WSDL message) or for a combination of these. The policies associated with a port are represented by a <PolicySet> element, that in turn can include other <PolicySet> elements representing the policies for an operation or a message. Each <PolicySet> element contains several <Policy> elements, which are associated with a single aspect of an end-point policy where an aspect is an independent set of technical features and parameters associated with the use of the Web service (for example, data rate allocation). The <Target> sub-element of a <Policy> element identifies the set of conditions governing the aspect (referred to as objective) of the end-point policy. Further, a <Policy> element must contain a set of <Rule> elements that define acceptable alternative solutions for achieving the objective. A <Rule> element includes a set of <Apply> elements containing predicates expressing conditions on attributes. Attributes can be of three different types: unconstrained, constrained and authorized. An unconstrained attribute is such that its value can be set by the policy-user, like for instance the minimum time between re-transmission of an unacknowledged message. The value of a constrained attribute, on the other hand, is out of the control of the policy-user. Examples of constrained attributes are environmental attributes like time, or subjects attributes the values of which are set by some an entity or user different from the policy-user (for instance, the status of the subject in a customer loyalty program). The value of an authorized attribute is asserted by an authority, like the policy-users role. Another interesting feature of XACML2 is that it adopts the XACML mechanisms for combining either multiple policies or multiple rules in a single policy, to blend the policies of the service consumer, expressing the preference/requirements of the consumer about the service provision, and the policies of the service provider. The <PolicySet> element, resulting from the combination process, represents a solution to both the consumer and provider policy statements. A service invocation using this solution conforms with the policy of both the consumer and the provider. This paves the way for the introduction of negotiation capabilities between the client and the service provider, that would ultimately increase the flexibility of the access control model. XACML2 is at the stage of a working draft.

Standards for Policy Specification The Web Services Policy Framework (WS-Policy) provides a framework that allows Web services to describe their policies to Web Service requestors. The main underlying concept is that a Web service provider might expose a policy to specify the conditions under which it provides the service, thus allowing the requestor to decide whether to use the service. The Policy Framework is supplemented by three other standards:

– WS-PolicyAssertions [18], that specifies a few generic policy assertions.

– WS-Policy Attachment [16], that defines how to associate a policy with a service, either by directly embedding it in the WSDL definition or by indirectly associating it through UDDI.
– WS-SecurityPolicy [17], that defines the security policy assertions corresponding to the security claims defined by WS-Security: message integrity assertion, message confidentiality assertion, and message security token assertion.

WS-Policy provides an extensible model and a single grammar for expressing all types of domain-specific policy models: transport-level security, resource usage policy, QoS characteristics, or the end-to-end business-process level policy. In WS-Policy, a policy is a collection of policy alternatives, where an alternative is a collection of policy assertions. The model and the grammar specify a core set of constructs to indicate how choices and/or combinations of domain specific policy assertions apply in a Web service environment.

The normal form schema of a policy according to Ws-Policy is shown in Figure 5. In this schema, * indicates 0 or more occurrences of an item, while [ ] indicates that the contained items must be treated as a group.

```
<wsp:Policy ... >
 <wsp:ExactlyOne>
     [<wsp:All> [<Assertion ...> ... </Assertion> ]* </wsp:All> ]*
 </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 5. Normal form schema of a policy according to WS-Policy

The assertions used in a policy expression can be defined in public specifications, like WS-SecurityPolicy [17] and WS-PolicyAssertion [18], or they can be defined by the entity owning the Web service. The assertions of the first type are named standard assertions and they are understandable potentially from any client. In particular, WS-SecurityPolicy specifies the security policy assertions that can be used in WS-Policy framework. The security policy assertions state requirements on the kind of security tokens used, whether or not a message has to be signed or encrypted. The assertions defined by the entity owning the Web service instead can be understood only by those clients to which the entity has already released the specifications. The policy assertions standardized so far are those defined in WS-SecurityPolicy, currently a public consultation draft release, and WS-PolicyAssertions. WS-Policy is also able to incorporate other policy models such as SAML and XACML. Figure 6 reports an example of policy that adheres to WS-Policy specification. This example, taken from the WS-Policy specification, shows two policy alternatives, each composed by a single policy assertion. The policy has to be interpreted as follows: if the first alternative is selected, only the Kerberos token type is supported; conversely, if the second alternative is selected, only the X.509 token type is supported.

```
<wsp:Policy xml:base=http://dico.unimi.it wsu:Id=MyPolicy>
    <wsp:ExactlyOne>
      <wsp:All>
        <wsse:SecurityToken>
            <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
        </wsse:SecurityToken>
      </wsp:All>
      <wsp:All>
        <wsse:SecurityToken>
           <wsse:TokenType>wsse:X509v3</wsse:TokenType>
        </wsse:SecurityToken>
      </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Fig. 6. An example of a policy

Defining the Web service policy is not however enough, if such a policy cannot made publicly available to the potential clients of the Web service when they try to discover the services they are potentially interested in. To this end, the WS-PolicyAttachment specifies mechanisms for using policy expressions with existing XML Web service technologies. In particular it defines how to associate policy expressions with WSDL [33] type definitions and UDDI [34] entities. It also defines how to associate implementation-specific policies with all or part of a WSDL portType when exposed from a specific implementation. WS-Policy and WS-PolicyAttachment are consortium public draft releases.

Security Management standards As described in the previous sections, keys and certificates are essential to guarantee the security of the exchanged messages. In the same way as standards are needed in order to provide interoperability between client and Web services, and among Web services, standards are needed to assure interoperability between a Web service and the certification authorities that issue, distribute and verify keys and certificates. This is the goal of XML Key Management Standard (XKMS) and WS-Trust standard.

XKMS - XML Key Management standard XKMS [19] specifies protocols for distributing and registering public keys, suitable for use in conjunction with the proposed standard for XML Signature and XML Encryption. XKMS comprises two services the XML Key Information Service (X-KISS) and the XML Key Registration Service Specification (X-KRSS). X-KISS allows a client to delegate part or all of the tasks required to process XML Signature <ds:KeyInfo> elements to an XKMS service. A key objective of the protocol design is to minimize the complexity of applications using XML Signature. By becoming a client of the XKMS service, the application is relieved of the complexity and syntax of the underlying Public Key Infrastructure (PKI) used to establish trust relationships, which may be based upon a different specification such as X.509/PKIX, SPKI or PGP. X-KRSS describes a protocol for registration and subsequent management of public key information. A client of a conforming service may request that the registration service bind information to a public key. The bound information may include a name, an identifier or extended attributes defined by the implementation. XKMS is a W3C Recommendation.

WS-Trust. WS-Trust [20] defines extensions to WS-Security that all together provide a method and a protocol for requesting to a Security Token Service the issuance, the renewal, and the validation of security tokens. The main goal of WS-Trust is to enable the issuance and dissemination of credentials among different trust domains. It intends to solve three potential issues that the recipient of a WS-Security-protected SOAP message faces when processing the security token contained within the Security header:

1. Format – the format or syntax of the token is not known to the recipient.
2. Trust – the recipient may be unable to build a chain-of-trust from its own trust anchors (e.g. its X.509 Certificate Authority, a local Kerberos KDC, or a SAML Authority) to the issuer or signer of the token
3. Namespace – the recipient may be unable to directly comprehend the set of claims within the token because of syntactical differences

In WS-trust terminology, a security token represents a collection of claims, where a claim is a statement made about a client, service or other resource (e.g. name, identity, key, group, privilege, capability, etc.). In particular, a signed security token is a security token that is cryptographically endorsed by a specific authority (e.g. an X.509 certificate or a Kerberos ticket). WS-Trust is an initial public draft release.

Identity Management standards Identity management (ID management) deals with identifying entities in a system (such as a country, a network, an enterprise, a Web service) and controlling their access to resources within that system. This is realized by associating rights and restrictions with the established identity. As an example, a Web service usually requests users to register, by asking name, address, and other information, and assigns them an account and a password. Then, to access the Web service the users must specify their identity, that is, their account and password; the Web service, in turn, uses these information to control the access to its resources. The management of identity-related information can be a costly process for Web service providers and might adversely impact both the security of the system and the users experience, particularly when the same user accesses several Web services The relevance of global identity management for the development of the Web and of the Web services, and the requirements to be addressed, were identified and discussed in a position paper by W3C, “Requirements for a Global Identity Management Service” [21]. The W3C position paper stipulates, among other things, that such a system that must be universally portable and interoperable; that it must support unlimited identity-related attributes; that it must provide adequate mechanisms for privacy and accountability; and that it must be overseen by an independent governing authority. One of the first application of Identity Management is Single Sign On (SSO). Single sign-on (SSO) technology enables users to perform a single action of authentication and allows them to authenticate themselves to all the services where they are eligible to, without the need to provide the same credential multiple times.

The other goal of ID management is to provide the means for building a federated identity infrastructure that enables cross-boundary single sign-on, dynamic user provisioning and identity attribute sharing. By providing for identity portability, identity federation affords end-users with increased simplicity and control over the movement of personal identity information while simultaneously enabling companies to extend their security perimeter to trusted partners. ID management is being investigated extensively in the corporate world and in standardization bodies and initiatives, such as W3C and Liberty Alliance. In what follows we describe the three major digital identity management initiatives.

WS-Federation. IBM/Microsoft WS-Federation [24] provides generic models for federated identity, attribute, authentication, and authorisation management, built upon WS-Trust and WS-Policy. A federation consists of multiple Web services domains, each equipped with its own Security Token Service, and with its own security policy. WS-Federation describes how to manage and broker trust relationships in a heterogeneous federated environment including support for federated identities, attributes, and pseudonyms. WS-Federation specifies scenarios using WS-Trust for example to allow requesters from the one domain to obtain security tokens in the other domain and subsequently to get access to the services in the other domain. Additionally, it defines mechanisms for SSO and single sign-out, sharing of attributes based on authorization and privacy policies, and integrated processing of pseudonyms (aliases used at different sites/federations). WS-Federation Specification is an initial public draft release and is provided for review and evaluation only.

Shibboleth. Shibboleth [25] is a project run by the Internet2 consortium in the USA. It is a standard-based, open source middleware software which provides a federated SSO across or within organizational boundaries, by implementing OASIS SAML 1.1 [22] specification. Shibboleth assumes that participating organizations have previously established a mutual trust relationship. It defines a protocol allowing users to access remote resources, where users are authenticated at their home site and authorized by a set of user attributes provided by their home site. The home site can use whatever type of authentication it likes e.g. username/password, Kerberos, digital signatures, and so forth. The users can be located at their home site, or anywhere else on the Internet. Under Shibboleth, the users home site is equipped with an Identity Provider (IdP) service, which maintains users credentials and attributes and that, upon request of relying parties, provides the relying parties with authentication or attribute assertions. Thus the IdP acts as an Authentication Authority. The remote Web site implements a Shibboleth Service Provider (SP) that manages the secured resources. When a user makes a request to access a remote Web site that does not know about him/her, the remote Web site redirects the user request to a Where Are

You From (WAYF) service, which maintains the list of the organizations whose users may access the resource. The user can choose his/her own home organization and then he/she is redirected to his/her home Web site, where he/she is authenticated. Then the IdP service running at his/her home site generates a handle for him/her, associates the authentication assertion with it, and redirects the users request back to the remote Web site. The service provider at the remote Web site validates the assertions received and, eventually, asks the IdP of the user home site additional assertions. These assertions are then passed to the Web application that, based on its access control policy, decides whether the users access is permitted or denied. Shibboleth preserves user privacy in that the IdP at the user home site, and the browser user can control what information is released to the service provider of the remote Web site. Thus, unless requested by the remote We site policy, users can remain anonymous and do not have to disclose sensitive identity information.

Liberty Alliance Project. The Liberty Alliance [26], [27] was formed in December 2001 to serve as an open standard organization for federated network identity management and identity-based services. The main concept of Liberty Alliance is to provide the specification of a federated identity infrastructure in which "individuals and businesses can more easily interact with one another, while respecting the privacy and security of shared identity information." [26]. Liberty Alliance is based on circle of trust concept. A circle of trust is constituted by service providers and IdP's. Service providers are organizations offering Web-based services to users. IdP's are service providers offering business incentives so that other service providers will affiliate with them. A circle of trust is created by establishing such relationships between service providers. Liberty Alliance architecture is organized around three building blocks: (1) the Federation Framework (ID-FF); (2) the Identity Web Services Framework (ID-WSF); (3) the Identity Services Interface Specifications (ID-SIS). ID-FF enables identity federation and management, and it supports, among the others, a simplified SSO and anonimity. ID-WSF is a foundational layer that uses ID-FF. Liberty ID-WSF defines a framework for creating, discovering, and consuming identity services. Identity Services Interface Specifications are a collection of specifications for interoperable services to be built on top of Liberty's ID-WSF. Liberty Alliance is based on SAML and provides open standards for SSO with decentralized authentication. In Liberty, a user authenticates via an identity provider, which may be any Internet service provider. As in Shibboleth, the actual method of authentication is not specified, and identity providers may use whatever authentication method they deem to be appropriate e.g username/password, X.509 public key certificates, Kerberos, one-time password schemes, and so forth. Identity federation is based on linking users service provider and IdP accounts. Each user has a different login identity at each service provider site. These identities are linked together by the underlying identity federation, but at each site, the user still continues to use his/her site-specific login identity. Login identities are not exchanged among the sites; they are referenced by a Liberty user handle, an identifier known by both sites and unique within the circle of trust. A Liberty handle is created by performing a hash of the users login identity and other information known only to the provider. Because the handle is at least 128 bits, it is virtually guaranteed to be unique. Note that each handle is only known by the two sites that are federated together, and a service provider will use different handles for the same user with different service providers. The user is in charge of federating sites together, and new handles are created each time, so that multiple sites cannot exchange information about the same user by using one handle. A vulnerability of this approach is that if a malicious user successfully authenticates itself once with the identity provider, it can subsequently use any of the victims information in the federation.

2.4 Issues in the Application of the Proposed Standards

Web services security standards raises three main broad questions. The first question is related to the specific security threats posed by XML itself and by the Web services security standards. The second one is related to the degree of interoperability that can be really achieved by the adoption of Web service security standards. The third one concerns the bandwidth and computational overhead that can be generated by the joint use of XML and of software packages implementing the Web services security standards.

With respect to the first question, XML by itself introduces new security threats like SQL injection through XML payload, XPath Injection, unexpected attachments, malformed XML and so forth. As to the enhanced security achievable by the adoption of security standards, we can say that the framework and the syntax defined by each specific Web service security standard do not provide by themselves any guarantee of security; each standard specification contains a specific section (Security Considerations) that describes the security concerns that the adopter of the standards should be aware of. These security concerns vary depending on the purpose of the standard. For example, when adopting security standards for SOAP messages encryption and signature, it must be taken into account that the XML signature created to verify data integrity is often sent in plain text. The plain text signature could be used by an attacker to perform an offline guessing attack potentially allowing the attacker to guess the value of the body of the message. Thus attention must be paid to use the Ws-Security feature that Encrypt also the SOAP signature.

As to the interoperability issue, the framework for security standard development identified in [2] postulates a layered approach, such that every upper layer standard can re-use and extend the specification of lower-layer standards. However, the specifications of the standard at a given layer (for example WS-Trust) are sometimes developed by a standardization body different from that specifying the standard at the other layer (for example SAML). Thus, the two involved standard specifications are not always compatible. Such situation requires an activity of verification and alignment of the specifications, that involves further iterations within each standardization body. Moreover, such an alignment might be further constrained by the fact that one of the standards involved is more stable and mature and is already implemented by some manufacturer.

A related issue concerns the real interoperability between the standard implementations by different manufactures. Although one of the main purposes of the standard is to guarantee the interoperability between different platforms, it might be necessary to test it on the field. This in turn might require a careful planning of the adoption and of the deployment of platforms implementing the standards.

As to the performance issue, the overhead induced by XML was addressed by the World Wide Web Consortium, that recently released three W3C Recommendations to improve Web services performance by standardizing the transmission of large binary data: XML-binary Optimized Packaging (XOP) [35], SOAP Message Transmission Optimization Mechanism (MTOM) [36] , and Reosurce Representation SOAP Header Block (RRSHB) [37]. These recommendations are intended to provides ways to efficiently package and transmit binary data included or referenced in a SOAP 1.2 message. The performance overhead generated by the processing requirements of the software implementing Web services security standards was tackled by the so-called XML appliances described in the next section.

2.5 XML Appliances

XML messages processing can require a very large amount of bandwidth with respect to traditional binary messaging protocols. Moreover, the processing of WS-* security compliant messages requires encryption/decryption and eventually signature management capabilities. Many manufacturers provide specialized products, based on proprietary hardware and operating systems, that have the objective of improving the performance of XML message processing and of providing an integrated management of XML-related security functions. Such products are commonly refereed to as XML appliances and include the XML accelerators and the XML firewalls. A XML accelerator appliance is a customized hardware and software where the following processing consuming tasks are performed: XML/SOAP parsing, XML schema validation, XPath processing and XSLT transformation functions. XML firewalls, also known as XML security gateways, are devices that, in addition to the functions of a XML accelerator, support the WS-Security standards and a range of security-related functions such as: content or metadata-based XML/SOAP filtering functions; XML messages encryption/decryption at the message or element level; XML signature verification and XML message signing according to XML Encryption standard; Authentication and authorization functions (that in some XML appliance can be based on local or on off-board repositories); Auditing and accounting functions.

Vendors of XML appliances such as IBM [38], Forum Systems [39] , Layer7 [40], Reactivity [41], Vordel [42], promote them as "a class of service intermediary" alternative to the approach of other vendors that enrich their middleware to provide a Service Oriented Architecture infrastructure.

3 Research Proposals and Open Research Issues

Despite such intense research and development efforts, current Web service technology does not yet provide the flexibility needed to "tailor" a Web service according to preferences of the requesting clients, thus failing to fulfil the mass-customization promises made at the beginning of the Web services era. At the same time, Web service providers demand enhanced adaptation capabilities in order to adapt the provisioning of a Web service to dynamic changes of the Web service environment according to their own policies. Altogether, these two requirements call for policy-driven access controls model and mechanisms, extended with negotiation capabilities. Models and languages to specify access and management control policies have been widely investigated in the area of distributed systems [43]. Standardization bodies have also proposed policy-driven standard access control models which we have surveyed in the previous section. The main goals of such models are to separate the access control mechanism from the applications and to make the access control mechanism itself easily configurable according to different, easy deployable access control policies. The characteristics of open Web environments, in which the interacting parties are mostly unknown each other, have lead to the development of the *trust negotiation* approach as a suitable access control model for this environment [48], [**?**]. Trust negotiation itself has been extended with adaptive access control, in order to adapt the system to dynamically changing security conditions. Automated negotiation is also being actively investigated in different application domains, such as e-business and Grid computing. However, a common key requirement that has been highlighted is the need of a flexible negotiation approach that enables the system to dynamically adapt to changing conditions. In addition, the integration of trust negotiation techniques with Semantic Web technologies, such as semantic annotations and rule-oriented access control policies, has been proposed. In such approaches, the resource under the control of the access control policy is an item on the Semantic Web, with its salient properties represented as RDF properties. RDF metadata, managed as facts in logic programming, are associated with a resource and are used to determine which policies are applicable to the resource. When extending a Web service with negotiation capabilities, the invocation of a Web service has to be managed as the last step of a conversation between the client and the Web service itself. The rules for such a conversation are defined by the negotiation protocol itself. Such a negotiation protocol should be described and made publicly available in a similar way as a Web service operation is publicly described through WSDL declarations. An XML-based, machine-processable negotiation protocol description allows an electronic agent to automatically generate the messages needed to interact with the Web service. Of course, the client and the Web service must be equipped with a negotiation engine that evaluates the incoming messages, takes decisions and generated the outgoing messages according to the agreed upon protocol. The models already proposed for peer-to-peer negotiations assume that both parties are equipped with the same negotiation engine that implements the mutually understood negotiation protocol. This assumption might not, however, be realistic and may prevent the wide adoption of negotiation-enhanced access control model and mechanisms.

In the remainder of this section we present a short overview of a system, addressing those requirements, and then we discuss open research issues.

3.1 Ws-AC1: an Adaptive Access Control System for Web Services

In order to address the adaption and negotiation requirements, that we have briefly outlined, a system has been recently proposed supporting a Web service access control model and an associated negotiation protocol [**?**]. The proposed model, referred to as Web service Access Control Version 1 (Ws-AC1, for short) is based on a declarative and highly expressive access control policy language. Such language allows one to specify authorizations containing conditions and constraints not only against the Web service parameters but also against the identity attributes of the party requesting the service and context parameters that can be bound, for example, to a monitor of the Web service operating environment. An additional distinguishing feature of Ws-AC1 is the range of object protection granularity it supports. Under Ws-AC1 the Web service security administrator can specify several access control policies for the same service, each one characterized by different constraints for the service parameters, or can specify a single policy that applies to all the services in a set; in order to support such granularity we introduce the notion of service class to group Web services. To the best of our knowledge Ws-AC1 is the first access control model developed specifically for Web services characterized by articulated negotiation capabilities. A model like Ws-AC1 has important applications, especially when dealing with privacy of identity information of users and with dynamic application environments. In order to represent the negotiation protocol, an extension to the Web Services Description Language (WSDL) standard has also been developed. The main reason of that choice is that, although the Web Services Choreography Description Language (WS-CDL) is the emerging standard for representing web services interactions, WS-CDL is oriented to support a more complex composition of Web services in the context of a business process involving multiple parties.

Ws-AC1 is an implementation-independent attribute based access control model for Web services, providing mechanisms for negotiation of service parameters. In Ws-AC1 the requesting agents (also referred to as *subjects*) are entities (human being or software agents) the requests by which have to be evaluated and to which authorizations (permissions or denials) can be granted. Subjects are identified by means of identity attributes qualifying them, such as name, birth date, credit card number, and passport number. Identity attributes are disclosed within access requests invoking the desired service. Access requests to a web service (also referred to as *provider agent*) are evaluated with respect to access control policies. Note that in its initial definition, Ws-AC1 does not distinguish between the Web service and the different operations it provides, that is, it assumes that a Web service provides just a single operation. Such model can be applied to the various operations provided by a Web service without any extension. Access control policies are defined in terms of the identity attributes of the requesting agent and the set of allowed service parameters values. Both identity attributes and service parameters are further differentiated in mandatory and optional ones. For privacy and security purposes, access control policies are not published along with the service description but are internal to the Ws-AC1 system. Ws-AC1 also allows one to specify multiple policies at different levels of granularity. It is possible to associate fine-grained policies with a specific service as well with several services. To this end, it is possible to group different services in one or more classes and to specify policies referring to a specific service class, thus reducing the number of policies that need to be specified by a policy administrator. A policy for a class of services is then applied to all the services of that class, unless policies associated with the specific service(s) are defined. Moreover, in order to adapt the provision of the service to dynamically changing conditions, the Ws-AC1 policy language allows one to specify constraints, dynamically evaluated, over a set of environment variables, referred to as *context*, as well as over service parameters. The context is associated with a specific service implementation and it might consist of monitored system variables, such as the system load. The access control process of Ws-AC1 is organized into two main sequential phases. The first phase deals with the identification of the subject requesting the service. The second phase, executed only if the identification succeeds, verifies the service parameters specified by the requesting agent against the authorized service parameters.

The identification phase is adaptive, in that the provider agent might eventually require the requesting agent to submit additional identity attributes in addition to those originally submitted. Such an approach allows the provider agent to adapt the service provisioning to dynamic situations: for example, after a security attack, the provider agent might require additional identity attributes to the requesting agents. In addition, to enhance the flexibility of access control, the service parameter verification phase can trigger a negotiation process. The purpose of this process is to drive the requesting agent toward the specification of an access request compliant to the service specification and policies. The negotiation consists in an exchange of messages between the two negotiating entities in order to limit, fix, or propose the authorized parameters the requesting agent may use. The negotiation of service parameters allows the provider agent to tailor the service provisioning to the requesting agent preferences or, at the opposite, to enforce its own preferred service provisioning conditions.

3.2 Open Research Issues

Even though Ws-Ac1 provides an initial solution to the problem of adaptive access control mechanisms for Web services, many issues need to be be investigated. A first issue is related to the development of models and mechanisms supporting a comprehensive characterization of Web services, that we refer to as *Web service profiles.* Such a characterization should be far more expressive that conventional approaches, like those based on UDDI registries or OWL. The use of such profiles would allow one to specify more expressive policies, taking into account various features on Web services, and to better support adaptation.

The second issue is related to taking into account the conversational nature of web services, according to which interacting with real world Web services involves generally a sequence of invocations of several of their operations, referred to as *conversation*. Most proposed approaches, like Ws-AC1, assume a single operation model where operations are independent from each other. Access control is either enforced at the level of the entire Web service or at the level of single operations. In the first approach, the Web service could ask, in advance, the client to provide all the credentials associated with all operations of that Web service. This approach guarantees that a subject will always arrive at the end of whichever conversation. However, it has the drawback that the subject will become aware of all policies on the base of which access control is enforced. In several cases, policies need to be maintained confidential and only disclosed upon some initial verification of the identity of the client has bene made. Another drawback is that the client may have to submit more credentials than needed. An alternative strategy is to require only the credentials associated with the next operation that the client wants to perform. This strategy has the advantage of asking from the subject only the credentials necessary to gain access to the requested operation. However, the subject is continuously solicited to provide credentials for each transition. In addition, after several steps, the client may reach a state where it cannot progress because the lack of credentials. It is thus important to devise strategies to balance the confidentiality of the policies with the maximization of the service completion. A preliminary approach to such strategies has been recently developed [28]; the approach is based on the notion of *k-trustworthiness* where k can be seen as the *level of trust* that a Web service has on a client at any point of their interaction. The greater the level of trust associated with a client, the greater is the amount of information about access control policies that can be disclosed to this client, thus allowing the client to determine early in the conversation process if it has all necessary credentials to satisfy the access control policies. Such approach needs however to be extended by integrating it with an exception-based mechanism tailored to support access control enforcement. In particular, in a step-by-step approach, whenever a client cannot complete a conversation because of the lack of authorization, some alternative actions and operations are taken by the Web service. A typical action would be to suspend the execution of the conversation, ask the user to acquire the missing credentials, and then resume the execution of the conversation; such a process would require investigating a number of issues, such as determining the state information that need to be maintained, and whether revalidation of previous authorizations is needed when resuming the execution. A different action would be to determine whether alternative operations can be performed to replace the operation that the user cannot execute because of the missing authorization. We would like to develop a language according to which one can express the proper handling of error situations arising from the lack of authorization.

The third issue is related to security in the context of composite services; in such case, a client may be willing to share its credentials or other sensitive information with a service but not with other services that can be invoked by the intial service. To handle such requirement different solutions may be adopted, such as declaring the possible services that may be invoked by the initial service or associating privacy policies with the service description, so that a client can specify its own privacy preferences. Other relevant issues concern worklow systems. Such systems represent an important technology supporting the deployment of business processes consisting of several Web services and their security is thus crucial. Even though some initial solutions have been proposed, such as the extension of the WS-BPEL standards with role-based access control [47], more comprehensive solutions are required, supporting adaptive access control and sophisticated access control constraints.

Finally the problem of secure access to all information needed to use services, such as information stored by UDDI registries, needs to be addressed. To date solutions have been developed to address the problem of integrity through the use of authenticated data structures [44]. However work is needed to address the problem of suitable access control techniques to assre the confidentiality and privacy of such information and thus to support its selective sharing among multiple parties.

4 Concluding Remarks

In this paper we have presented an overview of relevant standards concerning security for Web services. Such standards encompass a large number of security-related aspects, such as access control and digital identity management, and thus provides a solid basis for the deployment of Web service technology. We have also briefly outlined a more innovative approach that is the first proposing a negotiation-based access control model, able to provide support for adaptation, and we have discussed some open research issues.

5 Acknowledgement

The work reported in this paper has been partially supported by NSF under the project “Collaborative Research: A Comprehensive Policy-Driven Framework for Online Privacy Protection: Integrating IT, Human, Legal and Economic Perspectives”.

References

1. SOAP Version 1.2 W3C Recommendation 24 June 2003.
2. Microsoft and IBM - Security in a Web Services World: A Proposed Architecture and Roadmap 01 Apr 2002.
3. IETF The SSL Protocol Version 3.0 November 18, 1996.

4. IETF RFC 2246 January 1999.
5. RFC 2401 Security Architecture for the Internet Protocol November 1998.
6. XML Encryption Syntax and Processing W3C Recommendation 10 December 2002.
7. XML-Signature Syntax and Processing W3C Recommendation 12 February 2002.
8. SOAP message security (SOAP Message Security 1.0 WS-Security 2004) OASIS Standard 200401, March 2004.
9. Web Services Secure Conversation Language (WS-SecureConversation) February 2005.
10. Web Services Reliable Messaging Protocol (WS-ReliableMessaging) February 2005 specs.xmlsoap.org/ws/2005/02/rm/ws-reliablemessaging.pdf.
11. OASIS eXtensible Access Control Markup Language 2 (XACML) Version 2.0 OASIS Standard, 1 Feb 2005.
12. OASIS XACML profile for Web-services Working Draft 04, 29 Sep 2003.
13. ISO 10181-3 Access Control Framework.
14. WS-PolicyAttachment.
15. Web Services Policy Framework (WSPolicy) Version 1.2 March 2006.
16. D. Box, et al, "Web Services Policy Attachment (WS-PolicyAttachment)," March 2006. (See http://schemas.xmlsoap.org/ws/2004/09/policy.)
17. Web Services Security Policy Language (WS-SecurityPolicy) July 2005 Version 1.1.
18. Web Services Policy Assertions Language (WS-PolicyAssertions) Version 1.0 December 18, 2002.
19. XML Key Management Specification (XKMS 2.0) Version 2.0 W3C Recommendation 28 June 2005.
20. Web Services Trust Language (WS-Trust) February 2005.
21. Requirements for a Global Identity Management Service A Position Paper from OneName Corporation for the W3C Workshop on Web Services, 11-12 April 2001, San Jose, CA USA.
22. OASIS Security Assertion Markup Language (SAML) 2.0 Technical Overview - Working Draft 03, 20 February 2005.
23. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 (oasis-sstc-saml-core-1.1) OASIS Standard, 2 September 2003.
24. Web Services Federation Language (WSFederation) Version 1.0 July 8 2003.
25. Shibboleth Architecture Technical Overview Working Draft 02, 8 June 2005 http://shibboleth.internet2.edu/shibboleth-documents.html.
26. Liberty Alliance Project - Introduction to the Liberty Alliance Identity Architecture Revision 1.0 March, 2003.
27. Liberty Alliance Project - Introduction to the Liberty Alliance Identity Architecture Revision 1.0 March, 2003.
28. M.Mecella, M.Ouzzani, F. Paci, E. Bertino. Access Control Enforcement for Conversation-based Web Services. *Proceedings of the 2006 WWW Conference*, Edinburgh, Scotland, May 23-26, 2006.
29. Microsoft Message Queuing (MSMQ)
30. IBM WebSphere MQ V6 Fundamentals
31. Sun Java System Message Queue
32. IBM The Enterprise Privacy Authorization Language (EPAL 1.1) - Reader's Guide to the Documentation.
33. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft 26-March-2004.
34. OASIS UDDIVersion 3.0.2 UDDI Spec Technical Committee Draft, Dated 2004-10-19.

35. W3C XML-binary Optimized Packaging W3C Recommendation 25 January 2005. Available at http://www.w3.org/TR/2005/REC-xop10-20050125/
36. W3C SOAP Message Transmission Optimization Mechanism - W3C Recommendation 25 January 2005. Available at: http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/
37. W3C Resource Representation SOAP Header Block - W3C Recommendation 25 January 2005. Available at http://www.w3.org/TR/2005/REC-soap12-rep-20050125/
38. WebSphere DataPower SOA Appliances - http://www-306.ibm.com/software/integration/datapower/
39. Forum Systems Sentry - http://forumsystems.com/papers/Sentry_Data_Sheet_Spring_2004.pdf
40. Layer7 SecureSpan Gateway - http://www.layer7tech.com/products/page.html?id=2
41. Reactivity. http://www.reactivity.com/products/index.html
42. Vordel - http://www.vordel.com/products/
43. N. Damianou ,N. Dulay, E. Lupu and M. Sloman. The Ponder Policy Specification Language. *Proceedings of the 2nd IEEE International Workshop on Policies for Distributed Systems and Networks*, 2001.
44. E. Bertino, B. Carminati, E. Ferrari. Merkle Tree Authentication in UDDI Registries. *International Journal of Web Service Research*, 1(2): 37-57 (2004)
45. E. Bertino, E. Ferrari , A.C. Squicciarini. X -TNL: An XML-based Language for Trust Negotiations. *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2003.
46. E.Bertino, A.C. Squicciarini, L.Martino, F. Paci. An Adaptive Access Control Model for Web Services. To appear in *International Journal of Web Service Research*, 2006 (to appear).
47. E.Bertino, J.Crampton, F.Paci. Access Control and Authorization Constraints for WS-BPEL. Submitted for publication.
48. T. Yu, M. Winslett, K. Seamons. Supporting Structured Credentials and Sensitive Policies through Interoperable Strategies for Automated Trust Negotiation. *ACM Transactions on Information and System Security*, Vol. 6, No. 1, February 2003.