dDelega: Trust Management for Web Services

Michele Tomaiuolo
Department of Information Engineering, University of Parma, IT

ABSTRACT

In the context of Web services, access control presents some interesting challenges, especially when services are exposed to a global audience, with users accessing them from different systems and under different security settings. A decentralized approach to access control, which can be applied to such open environments, is represented by Trust Management. In fact, it is based on the peer-to-peer delegation of access rights among users, also across organizational boundaries, without supposing a-priori the existence of trusted third parties in the system. This article presents dDelega, a Trust Management framework for SOAP-style and REST-style Web services, available as open source software and usable in different application scenarios. The framework allows users to create multiple levels of delegation of access rights for protected resources. It defines various certificates, for binding names, permissions and oblivious attributes to users, adhering to relevant standards, such as WS-Security, SAML and XACML.

Keywords: Access Control; Authorization; Digital Signature; Public Key Encryption; Security Management; Web Technologies; XML

INTRODUCTION

The availability of services hosted on the Internet and invoked through open standard protocols allows developers to create modular and adaptive applications. Applications conforming to Web services standards can select and compose services at runtime, crossing both organizational and technological boundaries. In this sense, a standard Service-Oriented Architecture brings many benefits and high promises, in particular from the point of view of improved interoperability among diverse systems. Consequently, the number of open platforms hosting and providing Web services is growing, together with the number of initiatives, platforms and languages aimed at simplifying the integration of existing heterogeneous systems.

With regard to Web services security, a number of protocols and formats have been standardized by various organizations such as W3C, WS-I, OASIS, etc. A basic way of achieving security for Web services is relying on a secure transport layer, typically HTTPS and TLS. However, a message-level security is required in the case of architectures in which intermediaries can manipulate messages on their way. This was the rationale for the definition of new specifications, such as WS-Security. By using the XML-signature and XML-encryption specifications, WS-Security defines a standard way to secure SOAP messages, independently from the underlying transport protocol. As far as the REST-style is concerned, the security model is not as highly-developed as the security model for SOAP.

Despite all the efforts, realizing access control for protecting resources and services in an open context is still a challenging task, for both systems developers and system administrators. Moreover, in a global environment, it is not realistic to assume that all authorized principals are known in advance and listed in an ACL or some kind of prefigured policy. Since the recent widespread adoption of the Internet in consumer markets, contacts among people are often fully digitalized and there is still no definite solution to the problem of identity management. Actually, there could be no body of knowledge to associate with a name and the simple idea of building an on-line, global database of names and personal profiles is obviously infeasible as a general solution.

Given such a new way people are using the Internet today, a scheme of authorization based on peer-to-peer trust relationships becomes relevant, because it provides a scalable and easily extensible model to protect a generic resource across organizational boundaries. A possible approach to access control in an open environment is based on decentralized Trust Management (TM), i.e., on the peer-to-peer delegation of access rights among users. In Trust Management systems, in fact, the administrator of local resources is considered as the ultimate source of trust about them, and is provided with means to carefully regulate the flow of delegated permissions. No a-priori trusted parties are supposed to exist in the system, in general. This can also represent the basic setting for improved interoperability among diverse systems, paving the way for the realization of federated security infrastructures.

Thus, Trust Management fits well the context of an open Service-Oriented Architecture, for example in the case of composed services or in the presence of intermediaries between the requesters and the resources. In fact, such problems become more complex when the use of workflows involves many layers of services, with the possible presence of intermediaries. In this scenario, at each level an agent is responsible for managing its workflow. It can possibly subdivide its complex task into sub-tasks and set up a negotiation process with some agents responsible for the execution of simpler Web services. From the perspective of this example, two main abstract roles can be distinguished: the Service Manager and the Workflow Manager. In a typical peer-to-peer architecture, each agent can play different roles at different times. Each Service Manager is associated to one or more Web services and is responsible for the interaction with them. The Workflow Manager has the goal of supporting its user in the process of building a workflow. It composes external Web services and monitors their execution. In this sense, the Workflow Manager assumes the role of the delegate agent in a delegation protocol. However, these delegations cannot come into effect unless they are associated with a corresponding delegation of privileges, for accessing needed resources and completing assigned tasks.

This article describes the topic of Trust Managment in the context of Web services and presents dDelega, a security framework for both SOAP-style and REST-style Web services that allows the distributed delegation of access rights among different services and clients. The article is organized as follows. The next section provides a literature review about the principles of Trust Management and Automated Trust Negotiation, the use of oblivious attributes for preserving privacy in Trust Negotiation protocols, some relevant examples of access control frameworks proposed for Web services. Then, the dDelega framework is introduced, together with some basic services exploiting its delegation mechanisms. Finally, a balance about this ongoing work is drawn.

BACKGROUND

Trust Management

Apart from the use of Certification Authorities as trusted third parties, which is a quite traditional approach to system security, other solutions are possible. In Trust Management (TM) systems, in particular, the manager of local resources is considered as the ultimate source of trust about them, and it is provided with means to carefully administer the flow of delegated permissions. No a-priori trusted parties are supposed to exist in the system, in general, as this would imply some “obliged choice” of trust for the user, and without choice there is no real trust.

A number of architectures have been proposed for TM, including the Simple Distributed Security Infrastructure (SDSI) introduced by Rivest and Lampson (1996), and the Simple Public Key Infrastructure (SPKI) introduced by Ellison et al. (1999). They start from the observation that what computer applications often need is not to get the real-life identity of keyholders, but to make decisions about them as users (e.g. to grant access to a protected resource or not). More appropriately, Trust Management systems focus on principals and authorization. In general, a principal is any entity that can be taken accountable for its own actions in the system. In systems relying on asymmetric cryptography, principals could also be said to “be” public keys; i.e., if each principal has its own public key, then the

principal can be identified directly through its own public key and rights to access system resources can be bound to the same key.
Names can play an important role in a TM system. They are typically defined by a principal in its own local namespace. However, they can still can be used on a global scale, if they are prefixed with the public key (i.e. the principal) defining them (Rivest and Lampson, 1996). Then, each principal can issue a Name Certificate to associate some name (in the issuer's namespace) with its intended meaning (either a public key or another name). A Name Certificate creates a name→subject bound and is defined as a 4-tuple: (issuer, name, subject, validity). There's no limitation to the number of keys which can be made valid meanings for a name. So in the end, a Name Certificate can be used to define a named group of principals. Li, Grosof and Feigenbaum (2003) interpret these named groups of principals as distributed roles, paving the way for a dRBAC (distributed Role-based Access Control) paradigm.
Besides the possibility to assign local names to users, as distributed roles, another basic concept of TM systems is the Authorization Certificate, meant to create a straight authorization→subject bound (Ellison et al., 1999). It is defined as a 5-tuple: (issuer, subject, authorization, delegation, validity). Through an Authorization Certificate, a manager of some resources can delegate a set of access rights to a trusted subject. On its turn, this newly empowered principal can issue other certificates, granting a subset of its access rights to other entities. When finally requesting access to a resource, the whole certificate chain must be presented.
Li, Grosof and Feigenbaum (2003) discuss the importance of Authorization Certificates. Even recognizing that Authorization Certificates can improve the flexibility and granularity in permission handling, authors argue that most use cases can be satisfied by using local names and Name Certificates, only. In their perspective, local names are the distributed counterpart of roles in RBAC frameworks.
Figure 1 shows how Name Certificates can be organized in a chain, for realizing the delegation of access rights through the connection of local names. Like roles, local names can be used as a level of indirection between principals and permissions. Both a local name and a role represent at the same time a set of principals, as well as a set of permissions granted to those principals. But, while roles are usually defined in a centralized fashion by a system administrator, local names, instead, are fully decentralized. This way, they better scale to Internet-wide peer-to-peer applications.

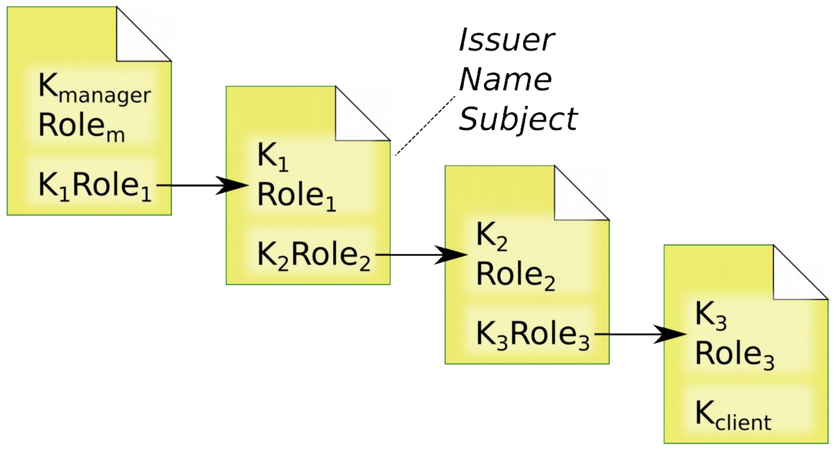

Figure 1. A delegation chain: $Role_m$ is granted indirectly from $K_{manager}$ to K_{client}

In this context, the delegation of goals and tasks is a key concept. But delegation is usually associated with a risk. And the decision of facing this risk is necessarily related to some form of trust. Trust is an important aspect of human life, and it has been studied under different point of views, for example in

the context of psychological and sociological sciences, or to draw specific economical models (Luhmann, 1979; Barber, 1959). In particular, Castelfranchi & Falcone (2003) define trust using a socio-cognitive approach. This way, it can be evaluated as a continuous function of its constituents, including competence, disposition, dependence and fulfilment. Following this approach, users and system administrators can take access control decisions on the basis of their local notion of trust, with respect to each one of their partners, without relying on external and possibly opaque security mechanisms and policies.
On the basis of a Trust Management model, Singh (2012) presents a method to rate authentication information as a level of trust. Moreover, a Quantitative Trust Management system is used to combine elements from both trust management and reputation management, while taking access control decisions.
It is worth noting that RBAC and dRBAC are established security models which do not necessarily lay on Trust Management principles. Gouglidis & Mavridis (2012), for example, present an extended RBAC model tailored to collaborative systems, named domRBAC. One of the main interesting features is the support for access control among participating domains. As usual, sessions are used as dynamic intermediaries between users and roles. Additionally, containers are used to represent both the environment and usage level information, to limit the usage of shared resources. Jin, Krishnan & Sandhu (2012) use a RBAC system for the assignment and modification of user attributes in an Attribute Based Access Control (ABAC) scheme. In fact, much research effort has focused on the problems of request evaluation and policy specification, assuming that users are somehow associated with a set of relevant attributes. Authors generalize the user-role assignment model of an RBAC system for handling more generic attributes.

Automated Trust Negotiation

Trust Management systems deals with the protection of local resources, realizing a distributed form of access control. However, also credentials used to create trust chains may contain sensitive data, including attributes, names and roles. This data needs to be protected at the same manner as other resources. Automated Trust Negotiation (ATN), in fact, enables interacting parties to establish trust incrementally, possibly starting as strangers and reaching some level of trust by disclosing credentials and policies iteratively (Seamons, Winslett & Yu, 2001; Yu & Winslett, 2003). Winsborough and Li (2000) discuss various negotiation strategies, ranging from an eager strategy, in which credentials are disclosed as soon as it is allowed by the access control policy, to a parsimonious strategy, in which credentials are disclosed only after ensuring that a successful result will be reached. Yu, Winslett and Seamons (2003) describe the Disclosure Tree as a family of strategies, which are proven to be interoperable with each other, i.e., interacting parties can choose each one a different strategy in the family and participate together to a negotiation. Winslett et al. (2002) introduce TrustBuilder as a modular system which can be used to perform customizable trust negotiation protocols. Lee and Winslett (2008) discuss the integration of TrustBuilder into a Service Oriented Architecture, in particular for realizing a Security Token Service which can carry on a Trust Negotiation conforming to WS-Trust specifications.
A number of cryptographic protocols have been proposed for solving possible stall situations in ATN. For example, protocols for solving cyclic dependencies in disclosure policies include Oblivious Signature Based Envelopes (Li, Du and Boneh, 2005), Hidden Credentials (Bradshaw, Holt and Seamons, 2004), Oblivious Commitment Based Envelopes (Li and Li, 2005), and secret handshakes (Balfanz et al., 2003). Other protocols, focused instead on the separation of credential disclosure from attribute disclosure, include Private Credentials (Brands, 2000), Anonymous Credentials (Belenkiy et al., 2009), and OACerts (Li and Li, 2005). Farràs et al. (2013) present a privacy-preserving mechanism for the exchange of policy and credentials during a trust negotiation process. Using secure multiparty computation techniques, client and server can obtain the list of matching policies and credentials,

respectively, without learning other private information. DeSPoT (Håvaldsrud et al., 2012) is a specific method for the development of policies for trust negotiation, mainly tailored to non-programmer domain experts, but useful in general for the configuration of a trust negotiation system.

Oblivious Attribute Certificates

An Oblivious Attribute Certificate (OACert) is a digital certificate which stores a cryptographic commitment of a user's attributes, instead of their values in the clear (Li & Li, 2005). It is an assertion about one or more attributes of the certificate holder, digitally signed by some authority. When the commitment system is secure, the certificate does not leak any information about sensitive attribute values. Since the protocol guarantees the separation of credential disclosure from attribute disclosure, the content of an OACert can be made public. This way, it is possible to solve some possible deadlocks during a Trust Negotiation and yet avoid the disclosure of sensitive information.

Through the disclosure of some OACerts, an authorized requester is able to satisfy the access policy of a service provider. In a typical zero-knowledge scheme, this happens without disclosing any sensitive information contained in the certificate attributes. Three types of entities participate in the protocol: (*i*) a trusted party T, responsible for certifying some attributes; (*ii*) a sender S, which is also a certificate holder; and (*iii*) a receiver R, which may be a service provider. The trusted party T is not necessarily bound to a hierarchical environment, as a X.509 Certification Authority, but it simply identifies an entity capable of issuing a certificate. Since the certificate alone does not allow anyone to gather information about the attribute values, the sender can show its OACert without having to worry about the privacy of its attributes. The generic scenario proceeds as follows: (*i*) each involved party owns a unique public/private key pair; (*ii*) a trusted party T generates an OACert for its future holder S; (*iii*) a receiver R, when presented with a request from a sender S, performs an access control on the basis of the attributes of S, certified in its OACert.

An OACert can be used in different protocols: (*i*) a protocol to open the commitment and thus reveal the attribute value; (*ii*) a Zero-Knowledge Proof protocol to prove that an attribute value satisfies a condition, without revealing more information; (*iii*) a protocol to guarantee that a requester obtains a resource only when its attribute values satisfy an access control policy.

In particular, the Oblivious Commitment Based Envelope (OCBE) protocol permits a requester to access a resource without revealing anything about the attribute values, not even about the effective satisfaction of the policy (Li & Li, 2005). It exploits various cryptographic primitives: (*i*) the Pedersen Commitment scheme, introduced by Pedersen (1991); (*ii*) a symmetric encryption algorithm, for example AES, such that a cleartext message can be encrypted and decrypted using the same key, $D_{key}(E_{key}(M)) = M$; and (*iii*) a one-way cryptographic hash function H, proven to be collision resistant.

For generating the commitments, the OCBE protocol exploits the Pedersen Commitment scheme, which can be summarized as follows. The scheme requires the generation of some system parameters at setup time by a trusted party T. Then, using those parameters, a sender S can generate a commitment for some value, and a receiver R can later verify the commitment.

- *Setup.* A trusted party T chooses two large primes p and q, such that q divides p – 1. T chooses g as a generator of G_q, the order-q subgroup of Z_p^* (={1, 2, …, p – 1}). T also chooses a random number a, with uniform probability, from Z_q (= {0, 1, . . . , q − 1}), and set $h = g^a \bmod p$. T then publishes (p, q, g, h) as the system parameters.
- *Commit.* To commit a value $x \in Z_q$, a sender S chooses a random number $r \in Zq$, with uniform probability. S then computes the commitment $c = (g^a h^r \bmod p)$. The domain of committed values is the finite field Z_q of q elements.
- *Open.* To open a commitment c, the sender S shows the values a and r. A receiver R can thus check whether $c = (g^a h^r \bmod p)$.

The Pedersen Commitment scheme is proven to be unconditionally hiding and computationally binding, based on the intractability of the discrete logarithm problem. On the basis of Pedersen Commitment scheme, OCBE is defined as a family of protocols for various predicates, including the basic comparison and logical predicates. As an example, the equality predicate (EQ-OCBE) will be sketched here, while other predicates can analysed in a similar fashion. The EQ-OCBE protocol is composed of various phases.

- *Parameter generation.* T runs a Pedersen commitment setup protocol to generate system parameters (p, q, g, h).
- *Commitment.* Given the value x to commit, T chooses a random value r and then computes the Pedersen Commitment $c = g^x h^r$. T sends x, r, c to R, and c to S. In practice, the commitment may be included into a certificate signed by T, and then communicated from the sender S to the receiver R.
- *Interaction.* Based on an equality predicate, representing the access policy, and the commitment c, the provider encrypts the protected resource and sends it to the requester, through an interactive protocol.
- *Open.* If the predicate is satisfied, R decrypts the resource using the secret value r.

Nabeel & Bertino (2013) present a privacy preserving attribute based group key management (PP AB-GKM) scheme, which combines the OCBE protocols with a Broadcast Group Key Management (BGKM) scheme. In this case, the rekey operation is performed by communicating new tokens on a broadcast channel. Each token needs to be combined with a user's secret to obtain the actual private key. Han et al. (2012) propose an oblivious access control scheme without zero-knowledge proofs. Their scheme is based on the Oblivious Signature Based Envelope (OSBE) protocol, but preventing credential holders to share their credentials with other users. OSBE is also shown to be viable as an Attributed-Based Encryption (ABE) scheme.

Access control frameworks for Web services

The Web Services access control is already becoming an important topic of many recent researches. The various security standards proposed and most of the studies carried out in the context of Web services focus mainly on the access control policies for single Web services (Bhatti et al., 2003; Bertino et al., 2006; Bhargavan et al., 2007). In particular, Bhargavan et al. (2007) address the problem of securing sequences of SOAP messages exchanged between Web services and their clients. By constructing formal models, they investigate the security guarantees offered by WS-Trust and WS-SecureConversation specifications. In particular they investigate the mechanisms allowing communicating parties to establish shared security contexts and to use them to secure SOAP-based sessions.

A few research works have dealt with security issues related to composed services. She, Thuraisingham and Yen (2007) propose a delegation-based security model to address problems such as how much privilege to delegate, how to confirm cross-domain delegation, how to delegate additional privilege. The proposed model extends the basic security models and supports flexible delegation and evaluation-based access control. But all Web services participating in this composition have to agree on a single token-based authorization mechanism, i.e., a hierarchical access control framework is provided. Bussard, Nano and Pinsdorf (2009) present a delegation framework which allows delegation of access rights in multi-domain service compositions. The approach is based on an abstraction layer, called abstract delegation, which harmonises the management of heterogeneous access control mechanisms and offers a unified user experience hiding the details of different access control mechanisms. dDelega differs from this approach because it considers each service or resource as a trust

domain based on a certificate chain access control mechanism.
The Grid Security Infrastructure (GSI) is the security layer realized for the Globus Toolkit (Welch et al., 2003). Differently from dDelega and other solutions based on Trust Management, the GSI essentially relies on X.509 Identity Certificates, issued by traditional authorities as trusted third parties. Additionally, GSI introduces X.509 Proxy Certificates as an extension to X.509 Identity Certificates. Those certificates are signed using a user's own credentials, instead of involving a CA, and allow the issuer to delegate some subset of his rights to some other entity created dynamically. Version 4 of GSI also uses SAML AuthorizationDecision assertions (Singh et al., 2011), in two ways: (*i*) SAML AuthorizationDecision assertions can be issued by the Community Authorization Service (CAS) to communicate the rights of CAS clients to services; (*ii*) the SAML AuthorizationDecision protocol allows to integrate an available authorization decision service, such as PERMIS.
PERMIS (PrivilEge and Role Management Infrastructure Standard) is an authorization infrastructure supporting both Role-Based and Attribute-Based Access Control (Chadwick et al., 2008). With PERMIS, administrators assign attributes and roles to users by issuing proper Attribute Certificates. The whole infrastructure also relies on traditional X.509 Certification Authorities for assigning public keys. Each service provider can use these attributes and roles to define the security policies regulating access to its own local services. To regulate access to a service, the local Policy Enforcement Point (PEP) asks a Policy Decision Point (PDP) for the list of required credentials. Other components involved in the operation are the Credential Issuing Service (CIS) and the Attribute Repository (AR), from which credentials can be collected. Policies are distinguished as Credential Validation Policies, to define the trusted authorities for each attribute, and Access Control Policy (ACP), to define the access rights associated with each attribute.
Shibboleth (Cantor, 2005) is a system for allowing the federation of security infrastructures deployed at various collaborating institutions, based on SAML formats and protocols. An important feature is the clear separation of the authentication process from the authorization process. The former is performed by an IdP (Identity Provider) service and assigns some attributes to an acknowledged user. The latter is performed by service providers on the basis of those proven attributes. This way, Shibbolet allows Single Sign On (SSO) among different sites and removes the need to maintain user names and passwords at those sites. Shibboleth is independent of the local authentication mechanisms, though they can be described through SAML assertions and later used for deciding about authorization. Shibboleth itself does not support the delegation of access rights among users and processes, but it has been integrated into other frameworks, including PERMIS and the Globus Toolkit for realizing grid applications. Additionally, interoperability with Information Card systems, such as CardSpace or Higgins, has also been demonstrated, through a browser extension (Al-Sinani & Mitchell, 2012).

THE DDELEGA FRAMEWORK

As a practical example about the use of distributed access control schemes in SOA-based applications, this section presents dDelega, a Trust Management framework for Web services. In particular, dDelega allows issuing and verifying chains of delegation certificates, to eventually associate a particular request for a service with some roles and permissions. At its core, it defines a hierarchy of classes for managing Name Certificates, Authorization Certificates and Oblivious Attribute Certificates. Certificates are encoded as SAML assertions, with the possible inclusion of XACML policies, as these languages are expressive, flexible and extensible. SAML and XACML are readily integrated into a Service Oriented Architecture, yet they may serve in different application scenarios. In fact, dDelega is the result of ongoing work started with the development of a security layer for JADE, one the most widespread FIPA-compliant multi-agent systems (Poggi, Tomaiuolo and Vitaglione, 2005). dDelega is distributed as open source software and is available at https://github.com/tomamic/dDelega.

dRBAC

Principals in dDelega are identified directly by their public key, similarly to other TM systems. This representation is also reproduced in SAML assertions. In fact, being designed to allow interoperability among very different security systems, SAML offers a variety of schemes for creating security assertions. In particular, it is possible to use a SubjectConfirmation element to represent a subject as the holder of a certain public key (which is a principal in a TM system). One of the main aims of dDelega is to implement a distributed RBAC access control system, along the lines discussed in (Li, 2000). For this purpose, local names are particularly important, as they allow each principal to manage its own namespace, which, on the other hand, is also one of the foundations of "federated identity" and SAML. In fact, while SAML allows the use of X.509 distinguished names, it also support a number of other heterogeneous naming schemes. In this sense, its reliance on XML provides intrinsic extendibility through schemas and namespaces.

In dDelega, assigning a local name to a public key, or to a set of public keys, is as simple as defining a role through a SAML assertion. In fact in SAML names and roles are not considered globally unique by design. And also assigning a named principal to a local name, or to a role, is perfectly possible. In particular, though not being foreseen in the specifications, a set of SAML assertions can also be organized into a certificate chain, like the one shown in Figure 1.

Ellison et al. (1999) note that, according to the X.509 PKI model, the issuer has the ability to eventually decide the conditions under which the certificate must be considered valid, and the enabled uses of the public key. But while the issuer must certainly be able to define the limits of its delegation, it is the final user who definitely takes a risk by accepting the certificate. Thus, if the relying party has to place some confidence in the certificate, it may need additional information about the assertion itself. In fact SAML allows the authentication authority to specify which mechanisms, protocols, and processes were used for the authentication.

Fine grained delegation

For using a SAML assertion to represent an Authorization Certificate, it is essential to have means for associating some access rights, or permissions, with the subject. In dDelega, this is achieved through the integration of an XACML policy into a SAML assertion. The precise way to accomplish this is described in a separate profile of the standard (Anderson and Lockhart, 2004). From the Trust Management perspective, the conjunction of SAML and XACML, in particular the inclusion of XACML policies and authorization decisions into SAML assertions, provides a powerful tool for the delegation of access rights.

From this point of view, the fact that logic foundations of the XACML language exist is very important, as they provide XACML with clear semantics. The problem is to find algorithms through which the combination of permissions granted in a chain of certificates could be computed in a deterministic way, as it is already possible in TM. In fact, even if the semantics of a XACML policy is logically sound, nevertheless subtle problems can appear when different policies, linked in a chain of delegation assertions, have to be merged. One major problem is about monotonicity of authorization assertions, which cannot be guaranteed in the general case. Using XACML authorization decisions as SAML assertions, it is possible to assert that access to a particular resource is denied, instead of allowed. Though being a perfectly legal and meaningful concept, the denial of a permission (a "negative permission") is not desirable in decentralized environments. In this case, a service provider can never allow access, as it cannot be sure to possess all issued statements. On the other hand, the non-monotonicity of the system can also lead to attacks, as issued assertions can be prevented to reach the provider, this way leading it to take wrong authorization decisions. Therefore, it is necessary to define a specific profile of SAML and XACML which could enable the secure delegation of permissions in decentralized environments, especially dealing with the case of "negative permissions".

Threshold subjects

In SPKI, threshold subjects are defined as a special kind of subjects, to be used only in Authorization Certificates. Li, Grosof and Feigenbaum (2003) question the usefulness of this construct, arguing it is used as an alternative to simulate conjunction and disjunction of subjects. Moreover, they provide an intuitive meaning for threshold subjects when used in Name Certificates. XACML does not support threshold subjects in their general case, but supports the conjunction of multiple subjects. In particular in dDelega it is possible to associate multiple subjects per access request, as the request could originate from a user, but it could also be mediated by one or more middle agents. The XACML Multi-Role Permissions profile specifies a way to grant permissions only to principals playing several roles simultaneously. In dDelega, this kind of policy can be defined by using a single Subject in its Target, but adding multiple Subject-Match elements to it.
Additionally, a Role Assignment policy can be used to define which roles are associated with which principals. Restrictions could be specified about the possible combinations of roles, thus limiting the total number of roles played by a principal. In principle, this way the disjunction of some roles could also be imposed. However, this use could be complicated in decentralized environments, as it could invalidate the monotonicity of the system. In fact, showing more credentials should never lead to obtaining fewer permissions.

Oblivious Attribute Certificates

Another feature of dDelaga is its support for Oblivious Attribute Certificates. The set of APIs for using OACerts inside dDelega is homogeneous with the rest of the framework and the whole Java environment. In particular, the package for OACerts, just like the other packages of dDelega, can be used in different contexts, including applications not based on Web services. OACerts are implemented as a subclass of the dDelega Certicate class, and represented as SAML assertions.
Various protocols based on OACerts are implemented. They are organized to guarantee easy of use and expansion. Two categories are defined, according to the protocol scope. The first group, descending from the VerifyScheme class, is meant to be used by a receiver, when it needs to match an oblivious credential against an access policy, before disclosing a resource. Two sublasses implement the DirectShow and ZeroKnowledge protocol. The second group, descending from the DecryptScheme class, allows (*i*) a provider to encrypt a resource to be sent to a requester; and (*ii*) the requester to decrypt the resource, if the access policy is satisfied. The initial implementation provides support for the EQ-OCBE (=) and GE-OCBE (≥) protocols.

Implementation remarks

The first step to develop a security infrastructure for Web services consisted in the realization of a core software library implementing basic functionalities, i.e., allowing the creation and validation of delegation certificates and certificate chains.
This core software library is designed in an abstract way, but it manipulates SAML and XACML structures under the hood. For managing basic SAML elements, dDelega exploits the OpenSAML library. In fact, it offers a good support for the latest SAML specifications and, above all, it allows the definition of new classes with relative simplicity. Extensibility is in fact particularly important, in this case, to realize a “glue” level between SAML and XACML, embodied by the XACMLPolicyStatement element. About XACML, the choice fell on Sun's XACML Implementation which is one of the most mature open source tool to deal with the language.
The core part of dDelega has a standard structure, exposing the API of a Java security provider. In fact, the Java Cryptographic Architecture (JCA) foresees the possibility to realize a package, called a security provider, to provide JDK with a concrete implementation of some cryptographic functionalities. In the case of dDelega, the main advantage of this choice is the availability of a set of APIs with a well known and practical structure. Moreover, this allows the use of certificate and path

classes available with normal Java APIs, without duplicating their functionalities. In principle, any component (also external ones) operating on a Java certificate, can perform basic operations on a certificate of the new library, too.
To realize an extension of the JCA, first of all it is necessary to extend basic Java data types, representing certificates and paths; then engine classes have to be realized, for specifying which algorithms are implemented. Finally, a master class for the provider has to be implemented, which is necessary to register new classes and allow them to be used by Java.
To represent certificates, JCA defines an abstract Certificate class which exposes a set of basic methods to manage public key certificates. Extending this class, dDelega defines a hierarchy of classes to represent Name Certificates, Authorization Certificates and Oblivious Attribute Certificates.
Ellison et al. (1999) describe an algorithm to evaluate the correctness of a certificate chain. To implement this validation algorithm and integrate it into the project, dDelega defines a subclass of CertPathValidator. Parameters of the validation process are represented as ValidatorParameters objects, containing the list of keys trusted by the principal operating the verification, and possibly additional parameters.
Another operation is the validation of an access request for a local resource. The request itself is represented by an instance of the AuthorizationRequest interface. Users of the framework can provide different implementations of the interface, according to their needs. Apart from the request, the algorithm must be provided with the list of Authorization Certificates to use and the list of locally trusted keys needed during the certificate verification process. Finally, in the case some additional conditions exist, it could be necessary to specify additional parameters for the verification process.
The validation happens through the creation of a Policy Decision Point (PDP). The Sun's XACML library provides the methods for creating such a decision block. To be able to obtain all needed policies, to validate the request, the PDP class of XACML uses various finder modules allowing to retrieve information. dDelega provides a finder module, called AuthzPolicyFinderModule, which is in charge of retrieving policies from authorization certificates provided as parameters.
During the process of creation of a PDP it is possible to insert additional finder modules. Such modules can be specified in the phase of construction of the AuthorizationEvaluator object and allow to extend the capabilities of the PDP to search for information. Moreover, this way it is possible to provide the validation module with a series of local policies which are not stored within Authorization Certificates.
The final result of the operation is a list of AuthorizationResponse objects, one for each requested resource. Each instance contains in its structure an identifier of the resource which it refers to, a decision value and a status code.

Experimentation setting

Some experimentations have been conducted with simple Web services, exploiting the security mechanisms of the dDelega framework, based on peer-to-peer delegations of access rights. In particular, a SOAP-based Security Token Service (STS) has been developed. It is responsible for creating a security session valid on a whole platform, so that a client can send his selected certificates just once, and then access a number of services provided on that platform. A quite similar service has also been developed according to the RESTful paradigm. A different STS supporting Automated Trust Negotiation (ATN) is also being implemented, by integrating the TrustBuilder framework with a different rule engine and some configurable strategies, and it will be included into the next version of the project.

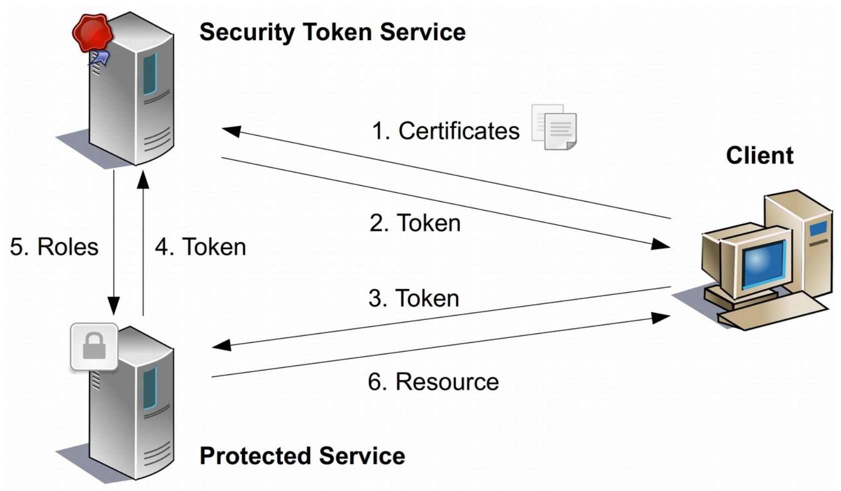

Figure 2. Experimentation scenario

SOAP services

The first sub-project of this experimentation has the objective of creating a peer-to-peer security mechanism for SOAP based services. The realized system is shown in Figure 2. It includes: (*i*) a Security Token Service (STS); (*ii*) a protected service, which needs proper authorization to be accessed; and (*iii*) a prototype client.

The STS is designed as a reusable service supporting the generic delegation of access rights among some peers. It allows a client to avoid attaching the same certificate chain to multiple service requests. The STS can accept and verify a certificate chain attached to a signed registration request. After a successful registration at the STS, the client is associated with a security session, which it can then specify when trying to access services on the same platform. The session token has to be obtained according to the WS-Trust specifications and it has to be used as a meaningful security token to be associated with WS-Security compliant messages.

The realized system effectively uses a number of technologies which have already been tested, and can work together to realize more complex scenarios than the ones foreseen in their specifications. In fact, the implementation is based on the Axis framework. Moreover all parties leverage Rampart to generate signed SOAP messages conforming to WS-Security specifications. All three parties are provided with their own couple of private and public keys, and can manage chains of delegation certificates and Oblivious Attribute Certificates encoded as SAML assertions.

The STS uses the dDelega framework for verifying a chain of delegation certificates or some OACerts. To effectively handle a registration request, it also verifies the signature of the request message and, if the certificate chain is correct, eventually generates a security session and returns a proper token to the client. The token can only be used in association with the public key used to verify the request message. The presence of the STS is made explicit through the security policy of the protected service, in conformance to WS-Policy specifications

The protected service verifies the proper authorization before granting access. For this purpose, it is associated with an Axis Handler which manages the session abstraction. The handler allows to separate the security aspect from the provision of the real service. Under the hood, it contacts the STS to receive a list of distributed roles associated with the public key and session token of the client, and then it uses an XACML policy to verify the association of the roles with the required permissions.

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

RESTful services

The second sub-project is in large part a replication of the first one, but in a RESTful environment. The main actors are still: (*i*) the STS; (*ii*) a protected service; and (*iii*) a client. The client needs to register at the STS, before accessing the protected service. The protected service relies on the STS for its access control mechanisms (see Figure 2).

The architectural style of the whole system, though, adheres to the REST model. Consequently, the different stack of involved protocols imposes some significant differences. First of all, the RESTful approach is much simplified with respect to the SOAP approach, as only plain HTTP messages are admitted and security is limited to TLS and HTTPS. For exploiting the specific features of the REST environment, this system also introduces some variations. First of all, the client is reduced to a plain web browser, which generates all requests and takes care of the cryptography. For this purpose, a private/public key pair (encoded into self-signed PKCS#12 certificates) has been installed in Firefox, enabling the not very usual policy of mutual authentication, allowed by HTTPS. Another difference is that the certificates are not sent directly in the request body, but as URLs of signed SAML documents, available as resources on the web. For the user, this simplifies the creation of the request message. Moreover, this scenario allows the automatic renewal of delegation certificates, by making the most recent version of a certain certificate always available in a well known location.

For realizing this experimentation, the Jersey framework has been used. While the internal operation of the STS remains essentially the same in both sub-projects, its API obviously changes for adhering to the chosen paradigm. Finally, the protected service uses an ad-hoc Filter, which takes care of contacting the STS and matching the acknowledged roles with the required permissions.

CONCLUSION

Besides traditional security models based on centralized or hierarchical Certification Authorities, other solutions are possible. Trust Management systems do not assume, a-priori, the existence of any globally trusted entity. This research work has shown that typical Trust Management schemes can be adapted to the context of Web services. Delegation of tasks and goals among services is a powerful technique for managing the complexity of modular and adaptive applications. These can run in open environments and rely on services provided under very different conditions.

However, delegation is also a complex process, which needs to be founded on a clear notion of trust. In fact, effective delegations of tasks often need to be associated with corresponding delegations of access rights. The risks of a delegation have to be taken into account, along with foreseen advantages. A possible approach is the socio-cognitive evaluation of trust toward a potential partner. It allows users to analyse the main constituents of a trust relation, including competence, disposition, dependence and fulfilment. This way, access control decisions can be founded on local beliefs, without having to rely on external mechanisms and policies.

An important feature of dDelega and other Trust Management systems is the possibility to use local names as distributed roles, on a global scale. This way, they allow users and system administrators to create trust relations across organizational boundaries, without forcing partners to adopt a unique security infrastructure. dDelega has the additional advantage of relying on standard formats and protocols for interoperability, conforming to Web services specifications. Thus, it is well suited for making the concept of linked local namespaces useful for the federation of existing security systems. In fact, the principle of delegation is not new and it is not a silver bullet, by itself. The interoperation among diverse mechanisms and policies has usually to overcome a number of serious issues. Those include technology mismatches, possibly caused by syntactic or semantic differences among policy and access control systems. Moreover, the heterogeneity often emerges from different underlying resource management and business models, and their background of culture, strategy and vision. Thus, at the very least, delegation among diverse security realms needs a common set of protocols and mediation mechanisms to overcome the heterogeneity of existing policies and mechanisms, without ambiguity. In

fact, a number of available standards and protocols, including SAML and XACML, can be adapted to a peer-to-peer Service Oriented Architecture. In particular, the organization of SAML assertions into a certificate chain enables the use of peer-to-peer delegation mechanisms in the context of different architectural styles. Since reimplementing all existing security infrastructures is often undesirable or simply infeasible, federated security is nowadays considered the key to build global security infrastructures, integrating already deployed security systems. This way, users are not obliged to adopt some out of the box solution for their particular security issues, to rebuild the whole system or to make it dependent upon some global authority, for gaining interoperability with others. Instead they're provided with means to manage the trust relations they build with other entities operating in the same, global environment. In particular, the idea at the basis of Trust Management is to make systems interoperate in the virtual world, just in the same manner as people collaborate in the real world, i.e., on the basis of evolving trust relations.
Regarding available and usable software, the more tangible results of this work include: (*i*) dDelega, a generic open source Java framework for issuing and verifying delegations chains; (*ii*) support for creating oblivious attributes and conveying them through SAML-based certificates; (*iii*) a Security Token Service exposing both a SOAP and a RESTful interface; and (*iv*) additional prototype components to experiment with the realized mechanisms in a standard Web services environment.

REFERENCES

Al-Sinani, H. S., & Mitchell, C. J. (2013). Enabling interoperation between Shibboleth and Information Card systems. *Security and Communication Networks*, *6*(2), 219-229.

Anderson, A., and Lockhart, H. (2004). *SAML 2.0 profile of XACML*. Retrieved 2013-08-20 from http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf

Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., and Wong H. C. (2003). Secret Handshakes from Pairing-Based Key Agreements. In *IEEE Symposium on Security and Privacy*, pp. 180-196.

Barber, B. (1959). *The Logic and Limits of Trust*. Grammercy Press, New Brunswick, NJ.

Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., and Shacham, H. (2009). Randomizable Proofs and Delegatable Anonymous Credentials. In Advances in Cryptology - CRYPTO 2009, ser. *Lecture Notes in Computer Science* vol. 5677, Springer Berlin / Heidelberg, pp. 108-125.

Bertino E., Squicciarini, A. C., Paloscia, I., and Martino, L. (2006). Ws-AC: a fine grained access control system for web services. *World Wide Web*, 9(2), 143-171.

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

Bhatti, R., Joshi, J. B. D., Bertino, E., and Ghafoor, A. (2003). Access Control in Dynamic XML-based Web-Services with XRBAC. In *Proceedings of the First International Conference on Web Services*, Las Vegas.

Bradshaw, R. W., Holt, J. E., and Seamons, K. E. (2004). Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pp. 146-157.

Brands, S.A. (2000). *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. ISBN:9780262024914. MIT Press.

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

Cantor, S. (2005). *Shibboleth Architecture. Protocols and Profiles*. 10 September 2005. Retrieved 2013-08-20 from http://shibboleth.internet2.edu/shibboleth-documents.html

Chadwick, D. W., Zhao, G., Otenko, S., Laborde, R., Su, L., and Nguyen, T. A. (2008). PERMIS: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11), 1341-1357.

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., and Ylonen, T. (1999). SPKI certificate theory. *IETF RFC 2693*, September 1999.

Farràs, O., Domingo-Ferrer, J., & Blanco-Justicia, A. (2013). Privacy-Preserving Trust Management Mechanisms from Private Matching Schemes. *arXiv preprint arXiv:1308.2435*.

Freudenthal, E., Pesin, T., Port, L., Keenan, E., and Karamcheti, V. (2002). dRBAC: Distributed Role-based Access Control for Dynamic Coalition Environments. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*, 2002, pp. 411-420.

Gouglidis, A., & Mavridis, I. (2012). domRBAC: An access control model for modern collaborative systems. *Computers & Security*, *31*(4), 540-556.

Han, J., Susilo, W., Mu, Y., & Yan, J. (2012). New constructions of OSBE schemes and their applications in oblivious access control. *International Journal of Information Security*, *11*(6), 389-401.

Han, J., Susilo, W., Mu, Y., & Yan, J. (2012). Efficient oblivious transfers with access control. *Computers & Mathematics with Applications*, *63*(4), 827-837.

Håvaldsrud, T., Møller-Pedersen, B., Solhaug, B., & Stølen, K. (2012). DeSPoT: A Method for the Development and Specification of Policies for Trust Negotiation. In *Computer Science and Convergence* (pp. 93-104). Springer Netherlands.

Jin, X., Krishnan, R., & Sandhu, R. (2012, September). A role-based administration model for attributes. In *Proceedings of the First International Workshop on Secure and Resilient Architectures and Systems* (pp. 7-12). ACM.

Lee, A. J., and Winslett, M. (2008). Towards Standards-Compliant Trust Negotiation for Web Services. In *Proceedings of the Joint iTrust and PST Conferences on Privacy, Trust Management, and Security (IFIPTM 2008)*.

Li, N. (2000). Local names in SPKI/SDSI. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, IEEE Computer Society Press, pp. 2-15.

Li, N., Du, W., and Boneh, D. (2005). Oblivious Signature-Based Envelope. *Distributed Computing*, 17(4), pp. 293-302.

Li, N., Grosof, B. N. and Feigenbaum, J. (2003). Delegation logic: a logic-based approach to distributed authorization, *ACM Transactions on Information and System Security*, 6(1), 128-171.

Li, J., and Li, N. (2005). OACerts: Oblivious attribute certificates. In Proceedings of the 3rd Conference on Applied Cryptography and Network Security (ACNS), ser. *Lecture Notes in Computer Science* vol. 353, pp. 301-3017. Springer.

Luhmann, N. (1979). *Trust and Power*. Wiley, New York, NY.

Nabeel, M., & Bertino, E. (2013). Privacy Preserving Delegated Access Control in Public Clouds. *IEEE Transactions on Knowledge and Data Engineering*, *99*(1), 1.

Pedersen, T. (1991). Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In Advances in Cryptology — CRYPTO '91, ser. *Lecture Notes in Computer Science* vol. 576, Springer Berlin / Heidelberg, pp. 129-140.

Poggi, A., Tomaiuolo, M., and Vitaglione, G. (2005). A security infrastructure for Trust Management in Multi-Agent Systems. In Trusting Agents for Trusting Electronic Societies, ser. *Lecture Notes in Computer Science* vol. 3577, pp 162-179.

Rivest, R.L., and Lampson, B. (1996). *SDSI - A Simple Distributed Security Infrastructure*. September 15, 1996. Retrieved 2013-08-20 from http://people.csail.mit.edu/rivest/sdsi11.html

Seamons, K.E., Winslett, M., and Yu, T. (2001). Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Network and Distributed Systems Symposium*.

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

Singh, A. (2012). Runtime Processes and Trust Management Model in MANET and GRID. *International Journal of Engineering Research and Applications (IJERA)*.

Singh, D., Gupta, B., Acharya, B. M., & Hota, S. (2011). An authorization Framework for Grid Security using GT4. *International Journal of Computer Science Issues*, *8*(1).

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

Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L., and Tuecke, S. (2003). Security for Grid services. In *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, pp. 48-57.

Winsborough, W. H., and Li, N. (2000). Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pp. 88-102, IEEE Press.

Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., and Yu, L. (2002). Negotiating trust in the Web. *IEEE Internet Computing*, 6(6), 30-37.

Yu, T., Winslett, M., and Seamons, K. E. (2003). Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Transactions on Information and System Security*, 6(1), 1-42.