Trust management and delegation for the administration of Web services

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

ABSTRACT
The availability and adoption of open protocols allow applications to integrate Web services offered by different providers. Moreover, simple services can be dynamically composed to accomplish more complex tasks. This implies the delegation of both tasks and permissions. In fact, delegation is intertwined with some notion of risk, on the one hand, and trust, on the other hand. Well founded socio-cognitive models of trust may orient managers and system administrators to delegate tasks and goals to the most trusted entities, after conscious evaluation of risks and gains associated with the decision. This chapter presents different mechanisms and models which have been proposed for establishing secure delegations in open environments. They include Role-Based Access Control, Trust Management and Federated Identity. Complex frameworks and live systems have been realized according to these models. However, their administration remain a challenging task. Ongoing research works in various fields, such as Automated Trust Negotiation, promise to simplify the practical realization and configuration of delegation-based systems.

Keywords: Web service; service composition; trust; delegation; access control.

1. INTRODUCTION
The adoption of a Service-Oriented Architecture based on Web services has definitely many benefits, above all from the point of view of interoperability among diverse systems. Web services offer a paradigm based on self-contained components, with a public description of their interface. Thus, modular applications can be developed on the basis of services hosted on the Internet and invoked through open standard protocols. This way, it is possible to select and compose services at runtime, crossing boundaries among underlying technologies and among organizations. Consequently, the number of platforms hosting and providing open services is growing, together with the number of systems developed through the composition of simpler services. Through service composition, applications can combine simpler services together in order to provide a certain useful functionality, even if no single Web service can satisfy the functionality by itself. This trend has triggered several initiatives, platforms and languages to simplify the integration of existing heterogeneous systems. Despite all the efforts, however, the realization of service composition through manual configuration is still a challenging task, for both systems developers and system administrators.

The service composition problem becomes more complex when the use of workflows involves many layers of services. 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, composing external Web services and monitoring their execution. The Workflow Manager assumes the role of the delegate agent in a delegation protocol. However, the dynamic composition of services provided by many different sources can be a cumbersome task. Therefore, the aid of software tools is critical for building composite Web
services. Efforts for realizing automated or semi-automated workflow composition have been conducted in different branches of artificial intelligence, including planning and theorem proving. Those efforts generally assume that the description of each Web service includes its preconditions and postconditions. Thus, it is theoretically possible to compose a number of services, in such a way to satisfy the final user's own constraints and goals. Being the service descriptions public, this process of dynamic composition of services can also be automated, even without a precise preconfigured workflow. Preconditions play the role of local constraints for the planning process. It's easy to see that the problem of service composition in dynamic and open environments is quite complex. However, matching explicit preconditions and effects may represent only a part of the problem. Particular attention has also to be devolved to the management of delegation chains and underlying trust relations, which often remain implicit. In fact, apart from the formal correspondence of preconditions and effects with final requirements, it is important to acknowledge the aspects of security and trust. In the end, the delegation of tasks and goals cannot come into effect unless it is associated with a corresponding delegation of privileges, for accessing needed resources and completing assigned tasks.

This chapter describes the advantages and issues related to the use of delegation in the composition of Web services. The “Background” section starts with a brief description of protocols for REST-style and SOAP-style Web services. Though they provide mechanisms for service composition, some analysis is always required for determining trust relations, which need to be at the basis of delegation decisions. For managers and system administrators, it is important to delegate tasks and goals to trusted partners, evaluating risks and gains of each decision. Socio-cognitive models of trust can be used to better evaluate and construct such relations. The following section deals with “Delegation Models and Systems”. It describes the main models of integration of security policies and mechanisms defined in different domains, including Role Based Access Control, Trust Management and Federated Identity. Then some major systems based on delegation of access rights are discussed. A case study is presented in greater detail, regarding the development of a delegation library for Web services.

In this rapidly evolving context, some aspects are object of ongoing research. They are described in the “Future Research Directions” section. Among the most promising fields, which may simplify the practical realization of delegation based systems, Automated Trust Negotiation certainly deserves the attention of developers and administrators. Oblivious Attribute Certificates and other cryptographic techniques may greatly improve such negotiations, since they can be used to develop trust reciprocally, without disclosing sensitive attributes.

Finally, some concluding remarks are provided, about the status of the field and the open challenges for system administrators.

2. BACKGROUND
The benefits of Web services, and Service Oriented Architectures in general, are well known. In particular, the adherence to open protocols greatly improves the interoperability among diverse systems. The popularity of service-based systems is intertwined with the growing trend toward Cloud Computing, which basically considers computing and storage capabilities as a commodity. Both trends guide towards more simplified and efficient data exchange and system integration activities. Also, both the service-based componentization of business applications, and the availability of scalable computing platforms, push organizations to adopt the most effective solution for each block in the workflow of the whole system.

With regard to security, a lot of efforts have been devoted to develop adequate standards 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. The administration of individual web services requires the configuration of security mechanisms at various levels, possibly including TLS, WSDL and WS-Policy. Nevertheless, both in the REST-style and SOAP-style Web services, the focus is on individual Web services. In the context of a Service-Oriented Architecture, the access issues in composed services or in the presence of intermediaries between the requesters and the resources still deserve more consideration. In such complex cases, the careful management of the delegation flow is one of the main challenges for system administrators. While the management of delegation decisions may require configuring security policies and issuing appropriate certificates and security tokens, the delegation process at its roots needs to be founded on an explicit model of trust. In fact, open systems can be described as environments where independent entities cooperate or compete, each one persecuting its own different interests, yet collaborating to achieve common goals. 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. Both Luhmann (1979) and Barber (1959), just to take two famous examples, analyze trust as a social phenomenon. In particular, Luhmann argues that trust is a fundamental instrument to simplify life in human society, ending with the idea that human societies can exist only on the base of trust. Barber associates the idea of trust with some expectations about the future: about the persistence of social rules, about the technical competence of the partners, and about the intention of the partner of carrying out their duties, placing others’ interests before their own. On the other side, other researchers analyze trust mainly in its psychological forms. Deutsch (1962) describes trust in terms of personal beliefs and perceived, or expected benefits. Gambetta (2000) is the first to give a definition of trust which is more grounded in mathematics, as a “subjective probability with which an agent assesses that another agent or a group of agents will perform a particular action...”. This definition appears more useful than the previous ones in the context of computer systems. In fact it is founded on the mathematical concept of “probability”, and this makes trust a quantifiable concept.

Yet, as Castelfranchi & Falcone argue (2001), the definition of trust as a “subjective probability” hides too many important details, thus being too vague to be applied in real cases. Instead, they present trust using a socio-cognitive approach, providing a deep analysis of a party’s believes, and the way they can influence trust. In particular, they list the beliefs about competence, disposition, dependence and fulfillment as important components of trust in every delegation, even towards non-cognitive software service providers. Instead, delegation towards people, organizations and social entities requires the delegating entity to hold additional believes about willingness, persistence and self-confidence of the partner, at least about the specific domain of the delegation. Then, using the socio-cognitive approach, trust can be evaluated as a continuous function of its constituents (Castelfranchi & Falcone, 2003), more precisely of the certainty of its constituent beliefs. But, though trust is a continuous function, the decision to delegate is necessarily discontinuous in its nature. The delegating entity can just decide to delegate or don’t delegate, and this decision has to take into account not only the degree of trust, but even other factors. These factors, including the importance of the goal, the perceived risk of frustrating the goal, the increased dependence on the trustee, and all other costs or possible damages associated with the delegation, will all influence a threshold function which will be eventually compared with the degree of trust for deciding whether to delegate or not.

Following this approach, security is deeply intertwined with both the degree of trust and the threshold function. In fact, security can certainly influence positively the trust on the partner, especially if security includes auditing mechanisms, certifications and confidentiality at the transport layer, which can help to associate a principal with its own actions and social behaviors. An even stronger degree of trust can be achieved when social interactions are founded on “contracts”,
i.e. signed documents that will make some party responsible for its own actions against an authority, a trusted third party able to issue norms, and to control and punish violations.

On the other hand, security mechanisms can be useful to limit the costs of a failed delegation. For example, delegation often comes in the twofold aspect of delegation of duties (performing actions or achieving goals), and delegation of corresponding permissions (rights to access the needed resources). In this case authorization mechanisms can be used to grant to the delegated entity only a limited number of access rights to valuable resources, thus limiting the damage that could be received from a misbehaving partner. In this way, security can be useful to reduce the threshold, and thus it can make delegation possible in a larger number of cases.

Moreover, when proper authorization mechanisms are available, delegation can be modulated according to the degree of trust, starting from the delegation of a single action, granting only the smallest set of strictly needed access rights, up to the delegation of a full goal, without specifying a plan or some needed actions to achieve it, and providing access to the largest set of available resources.

3. DELEGATION MODELS AND SYSTEMS

This section presents various systems allowing delegation of access rights for easing the tasks of system administrators, especially when authorization issues cross organizational boundaries. In fact, cooperation and agreements among companies and institutions are making virtual organizations both a reality and a necessity. But they’ll never spring into success if existing technologies will not match their needs. Particularly in the case of knowledge-based organizations, the continuous production of valued information poses a heavy burden on system administrators, who have to deal with security issues at an increasing pace. A centralized approach to the administration of system security can hardly scale to the most complex and dynamic settings. Delegation of authorization to local resource managers can instead help to manage the complexity of the whole administration job. On the one hand, delegation can be used to empower directly interested people in securing the sub-systems they rely on and they know better. On the other hand, it can reduce the scope of a possible breach due to a policy misconfiguration or a violation due to a user's malicious behavior.

Yet, the principle of delegation is not new and it is not a silver bullet, by itself. In fact, 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.

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 basis of evolving trust relations.

This section provides an overview of some attempts in this direction, including distributed Role- and Attribute-Based Access Control, Trust Management principles, security protocols for the federation of the information systems of different organizations. The analysis of those systems is accompanied by hints about the practice of their administration, especially in the context of a Service Oriented Architecture.
3.1. RBAC model
For the administration of open and interconnected systems, delegation is a requirement. In fact, a system administrator is not necessarily entitled to know the internal structure and roles of a partner organization. In any case, it is difficult to propagate information about people involved in a certain project timely and across organizational boundaries. It is often more convenient to leave the duty of assigning local roles and access rights for local resources to a local system administrator, instead of relying on centralized and remote administration.
A widely accepted scheme defines authorization in terms of Role-Based Access Control (RBAC) (Sandhu, Coyne, Feinstein & Youman, 1996). In this scheme, which is quite conventional nowadays, permissions are not assigned directly to users, but to roles. Those roles are then assigned to users, according to their tasks. Hence users acquire their own permissions indirectly, as they are associated to assigned roles. This level of indirection separates the assignment of permissions to roles, and of roles to users. Thus it eases the management of access control by system administrators.
There are many variations of RBAC in the literature, but typically roles are structured hierarchically, with senior roles being at higher level than junior roles and thus extending their set of access rights. A basic RBAC model, often called RBAC96, is introduced by Sandhu et al. (Sandhu, Coyne, Feinstein & Youman, 1996). A RBAC infrastructure based on X.509 is proposed by Chadwick et al. (2003). It stores a user's roles into an attribute certificate. Among the systems aimed at the protection of Web services, X-RBAC is developed by Bhatti et al. as a framework for the specification of access control policies based on XML. Another role-based access control system for Web services is proposed by Feng et al. (2004), as an extension to RBAC96. These systems can be used to define policies specifically for complex scenarios. Nevertheless, practice has demonstrated that administration of large and distributed systems through a Role-Based Access Control (RBAC) approach may remain a challenging task (Ferraiolo, Kuhn & Chandramouli, 2007). Various alternatives, designed for use in larger systems, are defined as distributed extensions to RBAC (dRBAC). Other systems, based on Attribute-Based Access Control (ABAC), assign attributes to users, which can be certified by specific authorities and later verified against access control policies. These alternatives often use the idea of delegation of access rights (Bandmann, Dam & Firozabadi, 2002) for making the duties of systems administrators more manageable. In fact, delegation can subdivide and decentralize access control tasks among various entities. Delegation has been applied to a wide range of applications and it is often considered as an effective mechanism to enhance the scalability of a distributed system. Related to RBAC, a role-based delegation mechanism is described by Na & Cheon (2000). Another mechanism for role delegation is proposed by Zhang et al. (2001), as an extension to the RBAC96 model called RDM2000. Freudenthal et al. (2002) argue that dRBAC should add some new features to previous approaches:
• Third-Party delegations allow some entities to delegate roles in different namespaces. This mechanism, related to the “speaks for” relationship in the Taos system, does not add any new functionality, as the same results can be obtained using anonymous intermediate roles, but improves the expressiveness and manageability of the system.
• Valued attributes allow authorities to add attributes and corresponding numeric values to roles. This way, access rights for sensible resources can be modulated according to some attributes. The same result could be obtained by defining different roles for different levels of access rights, but this would multiply the number of needed roles.
• Finally, continuous monitoring is needed, to verify the actuality of trust relationships. Typically, this feature is based on a publish/subscribe protocol to advertise the status updates of relevant credentials, which can be either revocable or short-lived.

3.2. Trust Management model
In contrast with the traditional approach to system security, based on Certification Authorities as trusted third parties, other solutions are possible. Trust Management is a different approach to
distributed administration. In fact, Trust Management (TM) is based on the concept of local trust, local names and delegation of access rights through delegation chains. In particular, in TM systems, 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.

Following the approach of (Li, Grosof & Feigenbaum, 2000), a generic system can be described as a community of peers, where each node is able to play the role of a controller or a requester. If an entity ultimately controls access to resources, being able to authorize or refuse requests for their usage, it plays the role of a controller; if an entity requests to access resources controlled by other entities, it plays the role of a requester. To have a sound system, all these peers should adhere to the principles of trust management. In (Khare & Rifkin, 1997) these rules of thumb are summarized as:
1. be specific: “Alice trusts Bob” is a too vague concept; it has to be better quantified in expressions as “Alice trusts Bob to read file.txt in her home directory today”;
2. trust yourself: all trust decisions should be founded on sound, local believes; when possible, trusted third parties should be avoided, especially if their mechanisms and politics are not known;
3. be careful: even the best implementation can be violated if users behave superficially and expose reserved data.

Applying these rules requires each service provider to be described as an authority, responsible for protecting its local resources and for managing its trust relations. This modus operandi provides a solid ground for the definition of trust relations among providers and consumers of services. Distributed RBAC can be realized also on the basis of a TM system, if local names are interpreted as distributed roles (Li, 2000). Distributed authorization mechanisms are especially useful in the case of federated security systems, which can rely on a number of standardized technologies. In the case of Service Oriented Architecture, federation can be realized using various combinations of WS-* specifications, SAML, XACML, OpenID and OAuth.

A number of architectures have been proposed for TM, including the Simple Distributed Security Infrastructure (SDSI) introduced by Rivest & 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.

In a typical TM scheme, local names defined by a principal can be used on a global scale, if they are prefixed with the public key (i.e. the principal) defining them (Rivest & 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 \( \text{name} \rightarrow \text{subject} \) bound and is defined as a 4-tuple: \((\text{issuer}, \text{name}, \text{subject}, \text{validity})\). 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 & Feigenbaum (2003) interpret these named groups of principals as distributed roles, paving the way for a dRBAC (distributed Role-based Access Control) paradigm.

An example of a delegation chain, realized through local name certificates, is shown in Figure 1.
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 & 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 Role Based Access Control (RBAC) frameworks. In fact, Name Certificates can be organized in a chain to link local roles, for delegation of access rights. 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.

3.3. Federated identity management
Typical access to Web services on the Internet is controlled on the basis of some identity management. The possibility of Single Sign-On is a further improvement over this basic scheme, since a user can login at a site, and then use services provided at different sites, while preserving security requirements. Identity federation schemes conveniently decentralize identity management and login services among various identity providers (Shin, Ahn & Shenoy, 2004). Such federated identity services usually allow trusted service providers, also in different security domains, to verify sensitive attribute and login information about a requester. Thus, a service provider can grant or refuse access to local resources on the basis of a user's status and attributes. For the practical realization of identity federation systems, various standard specifications are available, including SAML (Security Assertion Markup Language) (Hallam-Baker & Maler, 2002), WS-Federation (Hondo, Maruyama, Nadalin & Nagaratnam, 2006) and Liberty ID-FF (Identity Federation Framework). Those standards define how to handle an opaque security token issued to a particular user. Thus, their usage is important for improving interoperability, while preserving a certain level of privacy.

When the overall system architecture include some form of service composition, with providers taking actions on behalf of a user, then providers need to possess and show required access rights. Moreover, these privileges should be associated with a particular session, varying according to the final user's attributes and roles. In such a context, management and delegation of access rights is an indispensable aspect of the system, to control access to final services from intermediaries that must receive and use appropriate privileges from a responsible principal. The cited specifications about identity management depend on a strong assumption of generic “trust” among providers. Instead, they do not detail delegation mechanisms for transferring privileges from a user, behaving as a principal in the system, to an intermediate service provider, accessing some resources on his behalf. In fact, they can be configured for issuing different security tokens to providers in different security domains, either using SAML assertion or tokens based on WS-Federation and WS-Trust. But basically they assume that known providers are in a Circle of Trust (CoT), and support for delegation mechanism is quite limited.
Nevertheless, basic standards for the federation of Web services platforms can pave the way for more complex systems, which may introduce more advanced delegation and distributed RBAC mechanisms. SAML assertions, for example, are very generic and extensible, and can be used to encode delegation of access rights among cooperating parties. These delegations may be orchestrated by some acknowledged and trusted certification authorities, which can sign the appropriate assertions for attesting delegation (Gomi, Hatakeyama, Hosono & Fujita, 2005), or they can occur in a completely distributed choreography, on the basis of local relations developed according to Trust Management principles. This latest case will be discussed at the end of this section, along with the description of the dDelega library.

3.4. Delegation-based systems

Among the examples of security systems allowing the distributed delegation of access rights, it's worth mentioning the Grid Security Infrastructure (GSI), which is the security layer realized for the Globus Toolkit (Welch et al., 2003). Differently from 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 must be signed using a user's own credentials, instead of involving a CA. They allow the issuer to delegate some subset of his rights to some other entity created dynamically. The Globus Toolkit normally stores the user's private key locally, in a file. Other users of the same local machine are prevented from accessing the private key. In fact, the core GSI system encrypts the key through a secret pass phrase. The pass phrase must be inserted by the user at runtime, for decrypting the file containing his private key and thus being able to use the GSI. As an alternative, the system may be configured for using smart cards. This way, a user's private key is not stored in the local file system and it is better protected against unauthorized access from a malicious user. Version 4 of GSI can also be configured for using SAML AuthorizationDecision assertions, in two ways: (i) SAML AuthorizationDecision assertions can be issued by the CAS (Community Authorization Service) to communicate the rights of CAS clients to services; (ii) the SAML AuthorizationDecision protocol can be used to integrate a third party 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. The administrator of a certain service provider can then use these attributes and roles, for defining the security policies regulating access to local services. The access control for a service is performed by the local PEP (Policy Enforcement Point), which asks a PDP (Policy Decision Point) for the list of required credentials. The PDP is provided as a standalone authorisation server. It is a Web Service implemented as a Java based application, running with an embedded Apache Axis2 container. It can be configured to provide different services, including authorization decision, credential validation, obligation enforcement. It handles three protocols: XACML, XACML over SAML, Ws-Trust and SAML. Other components involved in the operation are the CIS (Credential Issuing Service) and the AR (Attribute Repository), 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. Mahmud et al. (2010) present a security scenario involving a sensor network in an e-health environment. Security policies for the e-health scenario are designed and enforced through PERMIS. Policies are based on a classical RBAC model, using X.509 Attribute Certificates and configured as a PMI (Privilege Management Infrastructure).

Shibboleth (Cantar, 2005) is a system for realizing the federation of security infrastructures. Shibboleth has been integrated into other frameworks, including PERMIS and the Globus Toolkit for realizing grid applications. It is deployed at various collaborating institutions, based on SAML formats and protocols. An important feature is the clear between (i) the authentication process, that
is performed by an IdP (Identity Provider) service and assigns some attributes to an acknowledged user, and (ii) the authorization process, that is performed by service providers on the basis of those proven attributes. This way, it is possible to use Shibboleth for Single Sign On (SSO) among different sites, removing the need to maintain user names and passwords at those sites. Shibboleth is independent of the local authentication mechanisms, though they can be described in SAML assertions and later used for deciding about authorization. A Shibboleth IdP may be configured to use multiple authentication methods at one time. In a request message, a service provider may supply a list of acceptable SAML authentication mechanism. If at least one of those mechanisms is supported by the IdP, then it can be used. If acceptable authentication mechanisms are not specified by the service provider, then the mechanism configured as default is used. To realize the SSO functionalities, the user's session is taken into consideration before other evaluations. If such session exists and its active authentication mechanism matches the requirements of the service provider, then the login handler of the existing session is used. 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.

A number of other systems have been proposed, especially in the field of Web Services. In fact, access control in Service Oriented Architectures 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, Joshi, Bertino & Ghafoor, 2003; Bertino, Squicciarini, Paloscia & Martino, 2006; Bhargavan, Corin, Fournet & Gordon, 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 the WS-Trust and WS-SecureConversation specifications, which provide communicating parties with mechanisms to establish shared security contexts and thus to secure SOAP-based sessions. She, Thuraisingham & 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 & Pinsdorf (2009) present a delegation framework for the delegation of access rights in multi-domain service compositions. The approach is based on an abstraction layer, called abstract delegation, which harmonizes the management of heterogeneous access control mechanisms and offers a unified user experience hiding the details of different access control mechanisms. Other systems, including the case study which will be discussed in the next subsection, follow a different approach. In an open distributed system, in fact, each service or resource can be considered as a trust domain on its own, and access control may be enforced on the basis of chains of delegation certificates.

3.5. Case study: dDelega as a generic delegation library
This section discusses the main advantages and the main issues to face for realizing generic security mechanisms for Web services, based on peer-to-peer delegation. In particular, it presents dDelega, a generic security library distributed as open source software (available at https://github.com/tomamic/dDelega). It can be used for 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 an abstract Certificate class, extended by concrete classes representing 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 & Vitaglione,
dRBAC. In TM systems based on asymmetric cryptography, principals can be identified directly by their public key. This may also be obtained in SAML. In fact, being designed to foster interoperability among very different security systems, SAML offers a variety of schemes for creating security assertions. In particular, there are a number of possible ways to represent a subject, including a SubjectConfirmation object 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 name space, which, on the other hand, is also one of the foundations of “federated identity” and SAML. In fact, while it is possible to use X.509 distinguished names with SAML, it also supports 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, it is perfectly possible to organize some SAML assertions into a certificate chain.

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.** With regard to the use of SAML as a representation of an Authorization Certificate, it would be important to have access rights, or permissions, associated 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 & 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 a clear semantic. 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 semantic 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 the 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 permitted. 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 & 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 conjunction of multiple subjects is possible in dDelega. In particular 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 the associations between roles and principals. Restrictions can 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.** The set of APIs supporting Oblivious Attribute Certificates is homogeneous with the rest of the dDelega library 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 Certificate class, and represented as SAML assertions.

With respect to implemented protocols, they are organized to guarantee simplicity and extensibility. The protocols are divided in two categories, according to their scope. The first category, 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 according to the DirectShow or the ZeroKnowledge schemes. The second category 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.

### 4. Future Research Directions

While Trust Management efforts are related to the protection of local resources, it has to be noted that credentials may convey sensitive data, as well. This data, including attributes, names and roles, needs to be protected at the same manner as other resources. **Automated Trust Negotiation (ATN)** (Winsborough & Li, 2000), 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. Various negotiation strategies are possible, as discussed by Winsborough & Li (2000), ranging from an eager strategy, in which credentials are disclosed as soon as it is permitted 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 & Seamos (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.

Like credentials, policies may convey sensitive data and thus may need protection, too. The problem is discussed by Seamos, Winslett & Yu (2001) and by Yu & Winslett (2003). Winslett et al. (2002) introduce TrustBuilder as a modular system which can be used to perform customizable trust negotiation protocols. Lee & 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.

The various open issues of ATN, related to the protection of credentials and possible stall situations in the protocols, are addressed by a number of cryptographic protocols. For example, protocols for solving cyclic dependencies in disclosure policies include Oblivious Signature Based Envelopes (Li, Du & Boneh, 2005), Hidden Credentials (Bradshaw, Holt & Seamos, 2004), Oblivious Commitment Based Envelopes (Li & 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
In particular, to avoid the disclosure of sensitive information and help solving possible deadlocks during a Trust Negotiation, Li & Li (2005) present a family of protocols based on the concept of **Oblivious Attribute Certificate** (OACert). Instead of storing a user's attribute values in the clear, an OACert stores a cryptographic commitment of such attributes. Thus, the certificate alone does not allow anyone to gather information about the attribute values. Nevertheless, in a typical zero-knowledge scheme, OACerts may allow a certificate holder to satisfy some policy, disclosing some certificates but not the sensitive information associated with its attributes.

An OACert is an assertion about the certificate holder, digitally signed by some trusted party. Each OACert contains one or more attributes. 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. The sender can show its OACert without having to worry about the privacy of its attributes. An attribute value in an OACert can be used in different ways to (i) open the commitment and thus reveal the attribute value; (ii) prove that an attribute value satisfies a condition, using a Zero-Knowledge Proof protocol and without revealing more information; and (iii) guarantee that the requester obtains a resource only when its attribute values satisfy an access control policy.

5. CONCLUSION

Delegation of tasks and goals among services is a powerful technique for realizing complex applications. However, it is also a complex process, which needs to overcome interoperability issues, and 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. Among the available mechanisms for establishing secure delegations in open environments, this chapter has analyzed Role Based Access Control, Trust Management and Federated Identity. They are all quite different approaches, but they pave the way to some form of delegation and trust among interconnected systems. Trust Management makes trust decisions more explicit, and thus it offers a more clear model of delegation for system administrators. Various security frameworks and complex systems have been analyzed, underlying the adherence to a particular scheme of access control. The configuration of delegation based systems remains a challenging task, above all if delegation decisions may happen at runtime in an open environment. Thus, research conducted in the field of Automated Trust Negotiation should be followed with attention, as it may provide simplified and automated procedures for realizing future delegation based systems.

REFERENCES


Welch, V., Foster, I., Kesselman, C., Mulmo, O., Pearlman, L., Tuecke, S., Gawor, J., Meder, S.,


ADDITIONAL READINGS


Yu, T., Ma, X., & Winslett, M. 2000. Prunes: an efficient and complete strategy for automated trust

**KEY TERMS AND DEFINITIONS**

**Automated Trust Negotiation**: a negotiation process which allows unknown users to establish a level of trust in an incremental way through the exchange of credentials. Automated Trust Negotiation can be used to automate trust building, guiding the disclosure of credentials according to privacy policies and negotiation strategies. The process is typically aimed at accessing some resource or some service.

**Delegation**: a mechanism for assigning the responsibility for completing a task, or satisfying a goal, to another entity. The delegation process may cascade across several levels, and may require a corresponding delegation of access rights.

**Digital certificate**: a digital document in which a issuer attests, via a digital signature, the association of one or more attributes to an entity, defined as the subject of the certificate.

**Federated identity**: the means of linking a person's identity and attributes, stored across multiple distinct identity management systems. Often Federated Identity is associated with an implementation of the Single Sign-On (SSO) scheme.

**Role-Based Access Control (RBAC)**: an access control scheme in which permissions are not assigned directly to users, but to roles. Those roles are then assigned to users, according to their tasks. Hence users acquire their own permissions indirectly, as they are associated to assigned roles. This level of indirection separates the assignment of permissions to roles, and of roles to users. Thus it eases the management of access control by system administrators.

**Single Sign-On**: a mechanism for establishing trust about a user's authentication process, performed at a certain site, across multiple identity management systems, also owned by different organizations. In this sense, SSO is an aspect of a federated identity management system. In particular, it provides interoperability among the authentication mechanisms of different systems.

**Trust**: the perceived probability that another agent will perform a particular action. This probability may be evaluated as a continuous function of the certainty of its constituent beliefs. The constituents and their importance may vary, according to the different existing socio-cognitive models of trust.

**Trust Management**: a system for the symbolic representation, creation and management of social trust. In particular, trust can be used for controlling access to protected resources. A request for accessing a resource is accepted only if accompanied with sufficient credentials, according to a local policy. Trust, quantified as a set of access rights, can be further delegated to other agents, creating trust networks. A chain of credentials may be used to represent the trust flow, from the resource manager to the agent requesting access.