NEW DIRECTIONS IN ACCESS CONTROL

Sabrina De Capitani di Vimercati, Pierangela Samarati
Dipartimento di Tecnologie dell'Informazione
Università di Milano
26013 Crema - Italy
{decapita,samarati}@dti.unimi.it

Abstract Access control is the process of mediating every request to resources and data maintained by a system and determining whether the request should be granted or denied. Traditionally, the access control process is based on a simple paradigm with basic functionalities (e.g., simple authorization tuples), the access control rules are under the control of a single party, and relying on user's authentication. The emerging open-based scenarios make inapplicable traditional assumptions. In this paper we illustrate recent proposals and ongoing work addressing access control in emerging applications and new scenarios.

Keywords: Access control policies and languages, attribute and certificate-based authorizations, security policy composition, semantic-based access control

1. Introduction

Access control is the process of controlling every request to a system and determine, based on specified rules, whether the request should be granted or denied [20]. Traditionally, an access control system is based on a simple paradigm where access restrictions are represented by simple authorization tuples stating that a subject s can perform an action a on an object o. These access control rules are usually under the control of a single party and relying on user's authentication. However, the emerging open-based scenarios make inapplicable these traditional assumptions. Also, it is widely recognized that a well-understood model and a highly expressive language for access control are of paramount importance in today's global network environment. Many solutions have been proposed to increase expressiveness and flexibility of authorization languages [11, 14, 15, 17, 23]. However, even these rich approaches result limiting. First, current approaches to enforcing access control are

all based on monolithic and complete authorization specifications. This is a limitation in many situations where the restrictions to be enforced come from different input requirements, possibly under the control of different authorities, and where the specifics of some requirements may not even been known a priori. This situation calls for a *policy composition framework* by which different component policies can be integrated while retaining their independence. Second, traditional assumptions for establishing and enforcing access control regulations do not hold anymore, since the traditional separation between authentication and access control does not apply, and alternative access control solutions should be devised. Writing access control policies where both requesters and resources to be protected are pointed at via data identifiers and access conditions evaluated against their generic *properties/attribute* seems to be a solution that can be used in open environments.

The remainder of this paper is organized as follows. Section 2 addresses the problem of combining authorization specifications that may be independently stated. We describe the characteristics that a policy composition framework should have and illustrate some current approaches and open issues. Section 3 addresses the problem of defining an access control system in open environments such as the Internet. We present the main requirements that an access control system should satisfy and describe current approaches and open issues. Finally, Section 4 concludes the paper.

2. Policy composition

Traditionally, authorization policies are expressed and managed in a centralized manner: one party administers and enforces the access control requirements. In many cases however, policy control has to be decentralized. For instance, in distributed environments, there may be multiple, independent and geographically distributed *entities* (i.e., individuals, organizations, institutes, and so on) with authority to control access to their local resources. Each of these parties is responsible for defining access control rules to protect resources and each brings its own set of constraints. To address these issues, a *policy composition framework* by which different component policies can be integrated while retaining their independence should be designed. The framework should be flexible to support different kinds of composition, yet remain simple so to keep control over complex compound policies. It should be based on a solid formal framework and a clear semantics to avoid ambiguities and enable correctness proofs.

In the following, we first describe the different requirements that must be addressed for a successful development and use of a policy composition framework. We then illustrate the main characteristics of some proposals and present some open issues.

2.1. Requirements of a policy composition framework

A first step in the definition of a framework for composing policy is the identification of the characteristics that it should have. In particular, we have identified the following [6]:

- *Heterogeneous policy support.* The composition framework should be able to combine policies expressed in arbitrary languages and possibly enforced by different mechanisms. For instance, a datawarehouse may collect data from different data sources where the security restrictions autonomously stated by the sources and associated with the data are stated with different specification languages, or refer to different paradigms (e.g., open vs closed policy).

- *Support of unknown policies.* It should be possible to account for policies which may be not completely known or even be specified and enforced in external systems. These policies are like "black-boxes" for which no (complete) specification is provided, but that can be queried at access control time. Think, for instance, of a situation where given accesses are subject, in addition to other policies, to a policy P enforcing "central administration approval". Neither the description of P, nor the specific accesses that it allows might be available; whereas P can respond yes or no to each specific request. Run-time evaluation is therefore the only possible option for P. In the context of a more complex and complete policy including P as a component, the specification could be partially compiled, leaving only P (and its possible consequences) to be evaluated at run time.

- *Controlled interference.* Policies cannot always be combined by simply merging their specifications (even if they are formulated in the same language), as this could have undesired side effects. The accesses granted/denied might not correctly reflect the specifications anymore. As a simple example, consider the combination of two systems P_{closed}, which applies a closed policy, based on rules of the form "*grant access if* $(s, o, +a)$", and P_{open} which applies an open policy, based on rules of the form "*grant access if* $\neg(s, o, -a)$". Merging the two specifications would cause the latter decision rule

to derive all authorizations not blocked by P_{open}, regardless of the contents of P_{closed}. Similar problems may arise from uncontrolled interaction of the derivation rules of the two specifications. Besides, if the adopted language is a logic language with negation, the merged program might not be stratified (which may lead to ambiguous or undefined semantics).

- *Expressiveness.* The language should be able to conveniently express a wide range of combinations (spanning from minimum privileges to maximum privileges, encompassing priority levels, overriding, confinement, refinement etc.) in a uniform language. The different kinds of combinations must be expressed without changing the input specifications (as it would be necessary even in most recent and flexible approaches) and without ad-hoc extensions to authorizations (like those introduced to support priorities). For instance, consider a policy P_1 regulating access to given documents and the central administration policy P_2. Assume that access to administrative documents can be granted only if authorized by both P_1 and P_2. This requisite can be expressed in existing approaches only by explicitly extending all the rules possibly referred to administrative documents to include the additional conditions specified by P_2. Among the drawbacks of this approach is the rule explosion that it would cause and the complex structure and loss of controls of two specifications; which, in particular, cannot be maintained and managed autonomously anymore.

- *Support of different abstraction levels.* The composition language should highlight the different components and their interplay at different levels of abstraction. This is important to: *i)* facilitate specification analysis and design; *ii)* facilitate cooperative administration and agreement on global policies; *iii)* support incremental specification by refinement.

- *Support for dynamic expressions and controlled modifications.* Mobile policies that follow (*stick with*) the data and can be enriched, subject to constraints, as the data move.

- *Formal semantics.* The composition language should be declarative, implementation independent, and based on a solid formal framework. The need of an underlying formal framework is widely recognized and in particular it is important to *i)* ensure non-ambiguous behavior, and *ii)* reason about and prove specifications properties and correctness [16]. In our framework this is particular important in the presence of *incomplete* specifications.

2.2. Summary of current policy composition frameworks

Various models have been proposed to reason about security policies [1, 11, 13, 18]. In [1, 13] the authors focused on the secure behavior of program modules. McLean [18] proposed a formal approach including combination operators: he introduced an algebra of security which enables to reason about the problem of policy conflict that can arise when different policies are combined. However, even though this approach permits to detect conflicts between policies, it did not propose a method to resolve the conflicts and to construct a security policy from inconsistent sub-policies. Hosmer [11] introduced the notion of meta-policies (i.e., policies about policies), an informal framework for combining security policies. Subsequently, Bell [2] formalized the combination of two policies with a function, called *policy combiner*, and introduced the notion of *policy attenuation* to allow the composition of conflicting security policies. Other approaches are targeted to the development of a uniform framework to express possibly heterogeneous policies [3, 14, 15, 17, 23]. Recently, Bonatti et al. [6] proposed an algebra for combining security policies together with its formal semantics. Following Bonatti et al.'s work, Jajodia et al. [22] presented a propositional algebra for policies with a syntax consisting of abstract symbols for atomic policy expressions and composition operators. The basic idea of these two proposals is to define a set of policy operators used for combining different policies. In particular, in [6] a policy is defined as a set of triples of the form (s,o,a), where s is a constant in (or a variable over) the set of subjects S, o is a constant in (or a variable over) the set of objects O, and a is a constant in (or a variable over) the set of actions A. Here, complex policies can then be obtained by combining policy identifiers, denoted P_i, through the following *algebra operators*.

- *Addition* (+) merges two policies by returning their set union. For instance, in an organization composed of different divisions, access to the main gate can be authorized by any of the administrator of the divisions (each of them knows users who needs the access to get to their division). The totality of the accesses through the main gate to be authorized would then be the union of the statements of each single division. Intuitively, additions can be applied in any situation where accesses can be authorized if allowed by any of the component (operand) policies.

- *Conjunction* (&) merges two policies by returning their intersection. For instance, consider an organization in which divisions

share certain documents (e.g., clinical folders of patients). Access to the documents is to be allowed only if all the authorities that have a say on the document agree on it. Intuitively, while addition enforces maximum privilege, conjunction enforces minimum privilege.

- *Subtraction* ($-$) restricts a policy by eliminating all the accesses in the second policy. Intuitively, subtraction specifies exceptions to statements made by a policy and it encompasses the functionality of negative authorizations in existing approaches, while probably providing a clearer view of the combination of positive and negative statements. The advantages of subtraction over explicit denials include a simplification of the conflict resolution policies and a clearer semantics. In particular, the scoping of a difference operation allows to clearly and unambiguously express the two different uses of negative authorizations, namely *exceptions to positive statements* and *explicit prohibitions*, which are often confused in the models or requires explicit ad-hoc extension to the authorization form [19]. The use of subtraction provides extensible as the policy can be enriched to include different overriding/conflict resolution criteria as needed in each specific context, without affecting the form of the authorizations.

- *Closure* ($*$) closes a policy under a set of inference (derivation) rules. Intuitively, derivation rules can be thought of as logic rules whose head is the authorization to be derived and whose body is the condition under which the authorization can be derived. Example of derivation rules can be found in essentially all logic based authorization languages proposed in the literature, where derivation rules are used, for example, to enforce propagation of authorizations along hierarchies in the data system, or to enforce more general forms of implication, related to the presence or absence of other authorizations, or depending on properties of the authorizations [14].

- *Scoping restriction* ($\hat{}$) restricts the application of a policy to a given set of subjects, objects, and actions. Scoping is particularly useful to "limit" the statements that can be established by a policy and, in some way, enforcing authority confinement. Intuitively, all authorizations in the policy which do not satisfy the scoping restriction are ignored, and therefore ineffective. For instance, the global policy of an organization can identify several component policies which need to be merged together; each component policy

may be restricted in terms of properties of the subjects, objects and actions occurring in its authorizations.[1]

- *Overriding* (o) replaces part of a policy with a corresponding fragment of the second policy. The portion to be replaced is specified by means of a third policy. For instance, consider the case where users of a library who have passed the due date for returning a book cannot borrow the same book anymore *unless* the responsible librarian vouchers for (authorizes) the loan. While the accesses otherwise granted by the library are stated as a policy P_{lib}, blacklist of accesses, meaning triples (`user`, `book`, `loan`) are stated as a policy P_{block}. In the absence of the *unless* portion of the policy, the accesses to be allowed would simply be $P_{lib} - P_{block}$. By allowing the librarian discretion for "overriding" the black list, calling P_{vouch} the triples authorized by the librarians, we can express the overall policy as $o(P_{lib}, P_{vouch}, P_{block})$.

- *Template* (τ) defines a partially specified policy that can be completed by supplying the parameters. Templates are useful for representing partially specified policies, where some component X is to be specified at a later stage. For instance, X might be the result of further policy refinement, or it might be specified by a different authority.

To fix ideas and make concrete examples, consider a drug-effects warehouse that might draw information from many hospitals. We assume that the warehouse receives information from three hospitals, denoted h_1, h_2, and h_3, respectively. These hospitals are responsible for granting access to information under their (possibly overlapping) authority domains, where domains are specified by a scoping function. The statements made by the hospitals are then unioned meaning that an access is authorized if any of the hospital policy states so. In term of the algebra, the warehouse policy can be represented as an expression of the form $P_1\hat{}[o \leq \mathtt{O_{h_1}}] + P_2\hat{}[o \leq \mathtt{O_{h_2}}] + P_3\hat{}[o \leq \mathtt{O_{h_3}}]$, where P_i denotes the policy defined by hospital h_i, and the scope restriction $\hat{}[o \leq \mathtt{O_{h_i}}]$ selects the authorizations referred to objects released by hospital h_i.[2] Each policy P_i can then be further refined. For instance, consider policy P_1. Suppose

[1] A simple example of scoping constraint is the limitation of authorizations that can be stated by a policy to a specific portion of the data system hierarchy [15].

[2] We assume that the information collected from the hospitals can be organized in abstractions defining groups of objects that can be collectively referred to with a given name. Objects and groups thereof define a partial order that naturally introduces a hierarchy, where $\mathtt{O_{h_i}}$ contains objects obtained from hospital h_i.

that hospital h_1 defines a policy P_{drug} regulating the access to drug-effects information. Assume also that the drug-effects information can be released only if the hospital's researchers obtain a patient's consent; $P_{consents}$ reports accesses to drug-effects information that the patients agree to release. We can then express P_1 as $P_{drug}\&P_{consents}$.

2.3. Open issues

We briefly describe some open issues that need to be taken into consideration in the future development of a policy composition framework.

- Investigate different *algebra operators and formal languages* for enforcing the algebra and proving properties. The proposed policy composition frameworks can be enriched by adding new operators. For instance, an *application operator* could be added that allows to take into consideration a policy only if the associated conditions evaluate to true. Also, the influence of different rule languages on the expressiveness of the algebra has not been yet investigated in the proposed approaches.

- *Administrative policies and language* with support for multiple authorities. The proposed approaches could be enriched by adding administrative policies that define who can specify authorizations/rules (i.e., who can define a component policy) governing access control.

- *Policy enforcement.* The resolution of the algebraic expression defining a policy P determines a set of ground authorization terms, that define exactly the accesses to be granted according to P. Different strategies can be used to evaluate the algebraic expression for enforcing access control: materialization, run-time evaluation, and partial evaluation. The first one allows a one-time compilation of the policy against which all accesses can be efficiently evaluated and which will then need to be updated only if the policy changes. The second strategy consists in enforcing a run-time evaluation of each request (access triple) against the policy expression to determine whether the access should be allowed. Between these two extremes, possibly combining the advantages of them, there are partial evaluation approaches, which can enforce different degrees of computation/materialization.

- Incremental approaches to enforce *changes to component policies*. When a materialization approach is used to evaluate the algebraic expression for enforcing access control, incremental approaches [21] can be applied to minimize the recomputation of the policy.

- *Mobile policies.* Intuitively, a *mobile policy* is the policy associated with an object and that follows the object when it is passed to another site. Because different and possibly indipendent authorities can define different parts of the mobile policy in different time instants, the policy can be expressed as a policy expression. In such a context, there is the problem on how ensure the obedience of policies when the associated objects move around.

3. Access control in open systems

In open environments such as the Internet resource/service requesters are not identified by unique names but depend upon their *attributes* (usually substantiated by certificates) to gain accesses to resources. Basing authorization on attributes of the resource/service requester provides flexibility and scalability that is essential in the context of large distributed open systems, where subjects are identified by their characteristics. Attribute-based access control differs from the traditional discretionary access control model by replacing both the *subject* by a set of attributes and *objects* by descriptions in terms of available properties associated with them. The meaning of a stated attribute may be a granted capability for a service, an identity or a non-identifying characteristic of a user (e.g., a skill). Here, the basic idea is that not all access control decisions are identity-based. For instance, information about a user's current role (e.g., physician) or a client's ability to pay for a resource access may be more important than the client's identity.

As before, we first describe the different requirements that must be addressed by an attribute-based access control system. We then illustrate the main characteristics of some proposals and present some open issues.

3.1. Requirements of an attributed-based access control system

Figure 1 depicts the basic scenario we consider. We are given different parties that interact with each other to offer services. A party can act both as a server and a client and each party has *i)* a set of services it provides and *ii)* a *portfolio* of properties (attributes) that the party enjoys. Access restrictions to the services are expressed by policies that specified the properties that a requester should enjoy to gain access to the services. The services are meant to offer certain functionalities that depend on the input parameters supplied by its users. Often input parameters must fulfill certain conditions to assure correct behavior of a

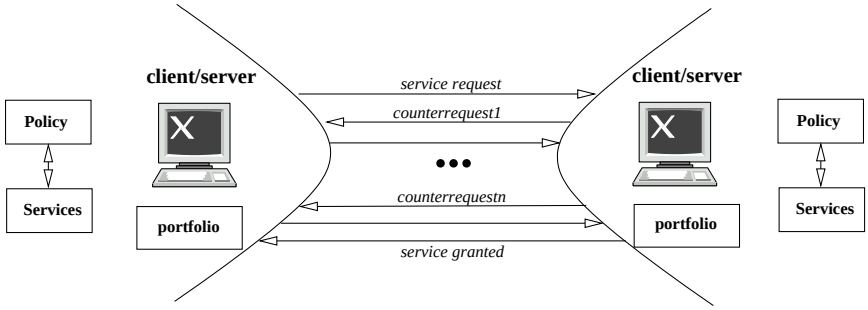

Figure 1. Client/server interaction

service. We identified the following requirements for specifying attribute-based access control for services.

- *Attribute interchange.* A server should be able to communicate to the client the requirements it need to satisfy to get access. Also, a client should be able to prove its eligibility for a service. This communication interchange could be performed in different ways (e.g., the involved parties can apply different strategies with respect to which properties are submitted).

- *Support for fine-grained reference to attributes within a credential.* The system should allow the selective disclosure of credentials which is a requirement that is not usually supported because users attributes are defined according to functional needs, making it easier to collect all credentials in a row instead of iteratively asking for the ones strictly necessary for a given service only.

- *Support for hierarchical relationships and abstractions on services and portfolio.* Attribute-based access control policies should be able to specify accesses to collection of services based upon collection of attributes processed by the requester.

- *Expressiveness and flexibility.* The system must support the specification of complex access control requirements. For instance, consider a service that offers telephone contracts and requires that the customer is at least 18 years of age. The telephone selling service has two input parameters, namely `homeAddress` and `noticePeriod`. The `homeAddress` must be a valid address in Italy and `noticePeriod` must be either one or three months. Further, the service's access control policy requires that contracts with one month notice period and home address outside a particular geographical region are closed only with users who can prove their

AAA membership. Hence, we see that the access control requirements of a service may require more than one interaction between a client and a server.

- *Support for meta-policies.* The system should provide meta-policies for protecting the policy when communication requisites. This happens when a list of alternatives (policies) that must be fulfilled to gain the access to the data/service is returned to the counterpart. For instance, suppose that the policy returned by the system is "citizenship=EU". The party can decide to return to the client either the policy as it is or a modified policy simply requesting the user to prove its nationality (then protecting the information that access is restricted to EU citizens).

3.2. Summary of current attribute-based proposals

To address the problems described in the previous section, some proposals have been developed that use *digital certificates*. Traditionally, the widely adopted digital certificate has been the *identity certificate*. An identity certificate is an electronic document used to recognize an individual, a server, or some other entity, and to connect that identity with a public key [4, 5, 8]. Identity certificates are certified by given entities (e.g., certification authorities). The certificate authority generally uses published verification procedures to ensure that an entity requesting a certificate is who it claims to be. When a certificate authority issues an identity certificate, it binds a particular public key to the name of the entity identified in the certificate (such as the name of a doctor). In addition to a public key, a certificate always includes additional information such as the name of the entity it identifies, an expiration date, the name of the certificate authority that issued the certificate, the digital signature of the issuing certificate authority, and so on.

More recent research and development efforts have resulted in a second kind of digital certificate, the *attribute certificate* [10] that can be used for supporting attribute-based access control systems. An attribute certificate has a structure similar to an identity certificate but contains attributes that specify access control information associated with the certificate holder (e.g., group membership, role, security clearance). Note that in principle these attributes can be placed in the extension fields of identity certificates [12]. However, this is not a viable solution for two main reasons. First, the certificate authorities who issue the identity certificates are not usually responsible for this kind of authorization information. As a result, certificate authorities must take additional steps

to obtain access control information from the source. Second, the lifetime associated with attribute-based information is different from the lifetime associated with identity-based certificates. In an attribute certificate, attributes need to be protected in a similar way to an identity certificate: they are therefore digitally signed sets of attributes created by *attribute authorities*. Attribute authorities are responsible for their certificates during their whole lifetime, as well as issuing them.

A first attempt to provide a uniform framework for attribute-based access control specification and enforcement was presented by Bonatti and Samarati in [7]. They propose a uniform framework for regulating service access and information disclosure in an open, distributed network system like the Web. Like in previous proposals, access regulations are specified as logical rules, where some predicates are explicitly identified. Attribute certificates are modeled as *credential expressions* of the form credential_name(attribute_list), where credential_name is the attribute credential name and attribute_list is a possibly empty list of elements of the form "attribute_name=value_term", where value_term is either a ground value or a variable. Besides credentials, the proposal also allows to reason about declarations (i.e., unsigned statements) and user-profiles that the server can maintain and exploit for taking the access decision. Communication of requisites to be satisfied by the requester is based on a filtering and renaming process applied on the server's policy, which exploits partial evaluation techniques in logic programs. Yu et al. [24, 25, 26] developed a service negotiation framework for requesters and providers to gradually expose their attributes.

3.3. Open issues

Although current approaches supporting attribute-based access control are technically mature enough to be used in practical scenarios, there are still some issues that need to be investigated in more detail to enable more complex applications. We summarize these issues as follows [7].

- *Ontologies.* Due to the openness of the scenario and the richness and variety of security requirements and attributes that may need to be considered, it is important to provide parties with a means to understand each other with respect to the properties they enjoy (or request the counterpart to enjoy). Therefore, common languages, dictionaries, and ontologies must be developed.
- *Access control evaluation and outcome.* Users may be occasional and they may not know under what conditions a service can be accessed. Therefore, to make a service "usable", access control mechanisms cannot simply return "yes" or "no" answers. It may

be necessary to explain why authorizations are denied, or - better - how to obtain the desired permissions. Therefore, the system can return an undefined response meaning that current information is insufficient to determine whether the request can be granted or denied. For instance, suppose that a user can access a service if she is at least eighteen and can provide a credit card number. Two cases can occur: *i)* the system knows that the user is not yet eighteen and therefore returns a negative response; *ii)* the user has proved that she is eighteen and the system returns an undefined response together with the request to provide the number of a credit card.

- *Filtering and renaming of policies.* As discussed above, since access control does not return only a "yes" or "no" access decision, but it returns the information about which conditions need to be satisfied for the access to be granted ("undefined" decision), the problem of communicating such conditions to the counterpart arises. To fix the ideas, let us see the problem from the point of view of the server (the client's point of view is symmetrical). The naive way to formulate a credential request, that is, giving the client a list with all the possible sets of credentials that would enable the service, is not feasible, due to the large number of possible alternatives. Also, the communication process should not disclose "too much" of the underlying security policy, which might also be regarded as sensitive information.

- *Negotiation strategy.* Credentials grant parties different choices with respect to what release (or ask) the counterpart and when to do it, thus allowing for multiple trust negotiation strategies [25]. For instance, an *eager* strategy, requires parties to turn over all their credentials if the release policy for them is satisfied, without waiting for the credentials to be requested. By contrast, a *parsimonious* strategy requires that parties only release credentials upon explicit request by the server (avoiding unnecessary releases).

- *Composite services.* In case of a composite service (i.e., a service that is decomposable into other services called component services) there must be some semi-automatic mechanisms to calculate the access control policy of a composite service from the access control policies of its component services.

- *Semantics-aware rules.* Although attribute-based access control systems allow the specifications of access control rules with reference to generic attributes or properties of the requestor and the

resources, they do not fully exploit the semantic power and reasoning capabilities of emerging web applications. It is therefore important to be able to specify access control rules about subjects accessing the information and about resources to be accessed in terms of rich ontology-based metadata (e.g., Semantic Web-style ones) increasingly available in advanced e-services applications [9].

4. Conclusions

Traditional access control models and languages result limiting for emerging Web applications. The open and dynamic nature of such scenario requires the development of new ways of enforcing access control. In this paper, we investigated recent proposals and ongoing work addressing access control in emerging applications and new scenarios.

5. Acknowledgments

This work was supported in part by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591 and by the Italian MIUR within the KIWI and MAPS projects.

References

[1] M. Abadi and L. Lamport. Composing specifications. *ACM Transactions on Programming Languages*, 14(4):1–60, October 1992.

[2] D.E. Bell. Modeling the multipolicy machine. In *Proc. of the New Security Paradigm Workshop*, August 1994.

[3] E. Bertino, S. Jajodia, and P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2):101–140, April 1999.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A.D. Keromytis. The role of trust management in distributed systems security. *Secure Internet Programming: Issues in Distributed and Mobile Object Systems. Springer Verlag LNCS State-ofthe- Art series,*, 1998.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. of the 1996 IEEE Symposiumon Security and Privacy*, Oakland, CA, USA, May 1996.

[6] P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, February 2002.

[7] P. Bonatti and P. Samarati. A unified framework for regulating access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[8] Y-H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. Referee: trust management forweb applications. *WorldWide Web Journal*, 2(3):706–734, 1997.

[9] E. Damiani, S. De Capitani di Vimercati, C. Fugazza, and P. Samarati. Extending policy languages to the semantic web. In *Proc. of the International Conference on Web Engineering*, Munich, Germany, July 2004.

[10] S. Farrell and R. Housley. An internet attribute certificate profile for authorization. RFC 3281, April 2002.

[11] H. Hosmer. Metapolicies ii. In *Proc. of the 15th National Computer Security Conference*, 1992.

[12] Information technology - open systems interconnection - the directory: Authentication framework, 2000. Recommendation X.509 (03/00).

[13] T. Jaeger. Access control in configurable systems. *Lecture Notes in Computer Science*, 1603:289–316, 2001.

[14] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(2):214–260, June 2001.

[15] S. Jajodia, P. Samarati, V.S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *Proc. of the 1997 ACM International SIGMOD Conference on Management of Data*, Tucson, AZ, May 1997.

[16] C. Landwehr. Formal models for computer security. *Computing Surveys*, 13(3):247–278, September 1981.

[17] N. Li, J. Feigenbaum, and B. Grosof. A logic-based knowledge representation for authorization with delegation. In *Proc. of the 12th IEEE Computer Security Foundations Workshop*, pages 162–174, July 1999.

[18] J. McLean. The algebra of security. In *Proc. of the 1988 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, USA, April 1988.

[19] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM TODS*, 16(1):89–131, March 1991.

[20] P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, LNCS 2171. Springer-Verlag, 2001.

[21] V.S. Subrahmanian, S. Adali, A. Brink, J.J. Lu, A. Rajput, T.J. Rogers, R. Ross, and C. Ward. Hermes: Heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes.

[22] D. Wijesekera and S. Jajodia. A propositional policy algebra for access control. *ACM Transactions on Information and System Security*, 6(2):286–325, May 2003.

[23] T.Y.C. Woo and S.S. Lam. Authorizations in distributed systems: A new approach. *Journal of Computer Security*, 2(2,3):107–136, 1993.

[24] T. Yu, M. Winslett, and K.E. Seamons. Prunes: Am efficient and complete strategy for automated trust negotiation over the internet. In *Proc. of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, November 2000.

[25] T. Yu, M. Winslett, and K.E. Seamons. Interoperable strategies in automated trust negotiation. In *Proc. of the 8th ACM Conference on Computer and Communications Security*, Philadelphia, PA, USA, November 2001.

[26] T. Yu, M. Winslett, and K.E. Seamons. Supporting structured credentials and sensistive policies trough interoperable strategies for automated trust. *ACM Transactions on Information and System Security (TISSEC)*, 6(1):1–42, 2003.