A Role-Based Access Control Model for Information Mediation *

Li Yang, Raimund K. Ege, Onyeka Ezenwoye, Qasem Kharma
School of Computer Science
Florida International University
Miami, FL 33199, USA
{lyang03|ege|oezen001|qkhar002}@cs.fiu.edu

Abstract

With the increasing demands for data integration and exchange among distributed heterogeneous sources, many applications require secure interoperation and the information sharing. Mediation techniques provide an extended amalgamation of searching and querying in heterogeneous systems, but enlarge the space of possible threats to local data sources. How to encourage data sharing while enforce required protection to resources is a challenging problem. Traditional access control mechanisms and methods are inadequate to reflect the heterogeneous environment and the flexible access control requirements. This paper presents a mediation security architecture for information integration based on role-based access control (RBAC). An adorned XML model (AXM) is used to homogenize security data modeling. Security requirements of mediation systems are specified by constraints over various RBAC dimensions. An incremental security enforcement method is proposed to integrate RBAC modules into the mediation architecture. The method supports adaptive and scalable design of secure mediation systems.

Keywords: *Mediation system, security, role-based access control*

1. Introduction

Information integration from many heterogeneous data sources is the trend for future information system. The mediation task is an extended amalgamation of searching and querying in traditional information systems [2]. Such task can be accomplished by the mediation strategies, i.e. semantic mapping [5, 3] and answering query by sources descriptions [16, 8]. Based on these mediation strategies, mediators are typically employed to provide integrated view of information from heterogeneous sources [1, 4]. A mediator provides a mapping of complex models to enable interoperability between clients and sources. With the promise of integrating data with rich semantics, we propose an adaptive mediator architecture [6, 18, 9] on the basis of the well-known mediator architecture [17]. The proposed architecture is reflective to the availability of sources and factors derived from the client and network capabilities. The mediators rewrite the request so that sub-request is tailored to access certain data source and the local data are streamed out to an integrated view.

However, mediation poses extensive security problems. In fact, protecting proprietary data from unauthorized access is recognized as one of the most significant barriers to the mediation in the information intensive fields, i.e., financial companies, hospitals. CHAOS[10] incorporates the security policies into the data objects as active nodes to form active objects. This model moves the responsibility of security to the source data provider, rather than through a central authority. Although the security managed by data sources provides convenience for the local sources, the mechanism prevent the mixed source information from leaking is void. The users can retrieve the unauthorized global information by integrating the authorized the local information. Traditional access control method such as mandatory access control (MAC) and discretionary access control (DAC) [11] are inadequate to reflect the flexible access control requirements in the dynamic mediator environment where collaborative access control is desired. Role-based access control (RBAC) [14, 7] models are receiving increasing attention as a generalized approach to access control. A role brings together a set of users on one side and a set of permissions on the other side. This greatly simplifies security management. This paper proposes and RBAC-based mediation security architecture. The mediator and security policies are specified respectively and the design towards implementation is discussed.

The rest of the paper is organized as follows: Section 2 proposes a mediation security architecture and explains me-

*This material is based upon work supported by the National Science Foundation under Grant No. HRD-0317692.

Copyright

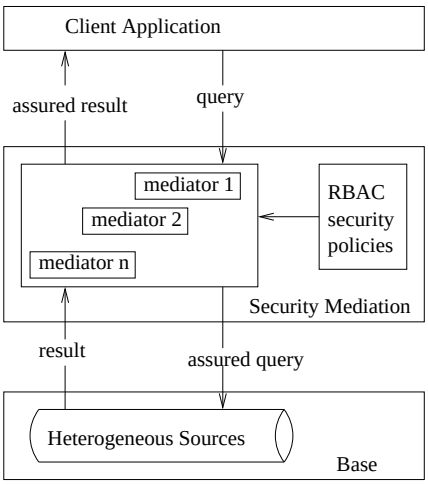

Figure 1. Security mediation architecture

diator specification. Section 3 specifies the security policies based on RBAC. Section 4 describes the security enforcement method. Section 5 is the conclusion.

2. Mediation Security Architecture

2.1. Overview

More and more individuals and organizations, whose databases could be heterogeneous, collaborate through mediation technique, to serve the client applications. Collaborators' access to their authorized information should be permitted, however, the access to the unauthorized information should be denied. For the safety reason, the latter is more important. Security mediators are designed to cope with security issues in collaborative computing environment. Mediators are intelligent middleware that sit between information system clients and sources. They perform functions such as integrating domain-specific data from multiple sources, reducing data to an appropriate level and restructuring the results into object-oriented structure. The mediators that are applied to security management are called security mediators. Security mediators interposes security checking between external accessors and data sources to be protected. They investigate queries incoming in and results to be transmitted to the external world. RBAC-based security specification is a rule system used to automate the process of controlling access and release of information. Applicable rules are combined to form security policies, which are enforced by the mediator for every user. Results are released only if they pass all tests. We formalize the role of the security mediation that has the responsibility and the authority to assure that no inappropriate information leaves an enterprise domain. Based on the criterions, i.e. expressibility, usability, support of new access right, adaptability, ease of administration, [20] analyzes and reasons that RBAC is the future direction to deal with the collaboration security. RBAC meets the natural needs of the organizations and enterprise management.

Figure 1 shows the conceptual architecture for mediation security where security specification is integrated with mediator to control access and release of information. The RBAC security policies part provide the high-level guideline from the aspect of (1) filter the mixed source information, (2) user assignment, (3) permission assignment and (4) session assignment. The mediator component specification language is Datalog and the security policies specification language is based on first-order logic. In logic view, the predicate of security specification can serve as the condition part of the Datalog specification in mediator, which makes it possible to regard the security specification as the rule systems for mediator specification.

2.2. Data Model

The primary motivation for mediation technology is to provide support for a broad spectrum of heterogeneous data which are available in different formats. A sound solution to the data integration task requires a clean abstraction of the different formats: any data must be mapped to an *exchange* model from which it is therefore accessible without the use of specific software.

We introduce a *lightweight* exchange model based on XML, enhanced via security (and potentially other) adornments. It is called *the adorned XML model* (AXM) [19]. AXM is flexible in data organization, both in the structures that can be described and in the differences in terminology. The security adornment of AXM is essential for the system security. Data represented in AXM is *self-description*. A second feature of AXM is flexibility in data organization, both in the structures that can be described and in the differences in terminology. The third feature of AXM is the security adornment that is essential for the system reliability. An AXM object has five attributes:

1. *Object ID*. It may be constructed by the mediators to be an expression describing where the object came from. It may also be a pointer to an object in the workspace used to answer the query.

2. *Label* tells what the object represents. Labels are expected to have human-understandable definitions that may be retrieved easily by the user.

3. *Adornment*. Adornment entry identifies the security properties that affect the data processing and system execution. Security adornment indicates a mapping

of principal identities and/or attributes thereof with allowable actions. It is a kind of security policy expression that is often essential in the access control in order to protect resources against unauthorized access. Security adornment plays an important role in the process by which use of resource is regulated according to a security policy and is permitted by only authorized system entities according to that policy.

4. *Type* of its value, either complex type or a simple type like *string*.

5. *Value*, either an atomic value or a set of objects.

With these primitives, it is possible to simulate all the structures that are found in more conventional object-oriented type systems. The adornments can be used not only to define the permissions of the objects in data sources but also to define the roles of the access user.

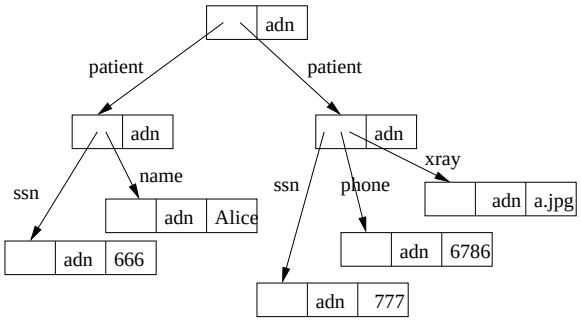

Figure 2. A collection of AXM objects

Figure 2 shows a collection of AXM objects. At the top is a *root* object whose label is *patientGroup*. Its value is a set of *patient* objects, so its type is *complex*. To model semi-structured information sources, we do not insist that data is as strongly structured as in standard database models. For instance, *name* information is sometimes given and sometimes missing and *address* could be a string or complex structure.

2.3. The Mediator Component Specification

In our mediation architecture the mediators in the integration layer form the components. Given a set of data sources exported from the homogenization layer, we build mediators to integrate and refine the information. The approach is in the spirit of the declarative specification of mediators in Tsimmis [13]. The query interpretation process is analogous to expanding a query against a conventional relational database view. We will use an example to illustrate the mediator specification and the query interpretation against the mediator specification.

Let us consider two mediators called *med* and *max* that export objects with label *patient*. The *patient* objects fuse information about patients that have the same social security number and are exported by the sources s_1, s_2 and s_3. In particular, if source s_1 contains a patient and his name, the exported *patient* object contains the corresponding *name*. If s_2 contains the x-ray examination for the patient and s_3 contains the address information, then the *xray* and *addr* sub-objects are also included in the *patient*. A specification consists of *rules* that define the view exported by the mediator. Each rule consists of a head followed by a $:-$ and a tail. The head describes view objects, whereas the tail describes conditions that must be satisfied by the source objects. In general, the heads and tails are based on patterns of the form $\{<$*object-id adornment label value*$>\}$.

The specification of the *patient* object appears in two mediators specification ("MS1" and "MS2"); each rule in the specifications describes the contribution of the sources:

(MS1) (R1.1)
<oid(S) (adn R) patient {<(adn R) name N>}>@med :−
<(adn P) patient {<(adn P) ssn S> <(adn P) name N>}>@s1
AND role_assign() AND check_permission()
(R1.2)
<oid(S) (adn R) patient {<(adn R) xray X>}>@med :−
<(adn P) patient {<(adn P) ssn S> <(adn P) xray X>}>@s2
AND role_assign() AND check_permission()

(MS2) (R2.1)
<oid(S) (adn R) patient {<(adn R) addr A>}>@max :−
<(adn P) patient {<(adn P) ssn S> <(adn P) addr A>}>@s3
AND role_assign() AND check_permission()
(R2.2)
<oid(S) (adn R) patient {<(adn R) xray X>}>@max :−
<(adn P) patient {<(adn P) ssn S> <(adn P) xray X>}>@s2
AND role_assign() AND check_permission()

The first rule declares that:

- **if** there is a pair of *binding s* and *n* for variables *S* and *N* (variables are identifiers starting with a capital letter) such that s_1 contains a *patient* top-level object that has a *ssn* sub-object with value *s* and a *name* subobject with value *n*,
- **then** mediator *med* exports a *patient* object, with object-id *oid(s)*, that has a *name* subobject with value *n* and a unique system-generated object-id.

The semantics of the second rule in *MS1* ("R1.2") and in the two rules in *MS2* are defined accordingly. Notice that how *patient* object at the mediator is assigned the semantic object id *oid(s)*. (We add the function symbol *oid* to the social security number obtained from the source to uniquely identify how this id was generated.) Observe that (*MS1.1*) does not prevent the *patient* with object id *oid(s)* to have subobjects other than *name*, thus allowing the second rule to add more subjects to the same *patient* objects.

To illustrate how to interpret the query against the mediator specification, assume that a client wants to retrieve all data of *patient's* with object-id *oid('666')*. The query can

be expressed as:

(Q1) <oid('666') (adn R) patient PT}> :−
<pid('666') (adn P) patient PT>

The object pattern (or patterns in the general case) that appears in the query tail is evaluated against the object structure of the mediator in exactly the same way that the mediator specification rule tails are evaluated against the object structures of the $mediator\_connector$. The object pattern of the query head does not include the usual "@" notation because it is implied that the objects described by the query head refer to the result that will be materialized by the client.

After evaluation the tail of sample query (Q1) against the head of the rules in *(MS1)* and *(MS2)*, (Q2), (Q3) and (Q4) are sent to the sources s_1, s_2 and s_3 respectively.

(Q2)
<oid('666') (adn R) patient {<(adn R) name N>}> :−
<(adn P) patient {<(adn P) ssn '666'> <(adn P) name N>}>@s1

(Q3)
<oid('666') (adn R) patient {<(adn R) xray X>}> :−
<(adn P) patient {<(adn P) ssn ''666> <(adn P) xray X>}>@s2

(Q4)
<oid('666') (adn R) patient {<(adn R) addr A>}> :−
<(adn P) patient {<(adn P) ssn '666'> <(adn P) addr A>}>@s3

The three answer objects received from s_1, s_2 and s_3 are then merged into a single *patient* object.

3. Mediation Security Specification

3.1. An Introduction to RBAC Models

RBAC is a well-defined research area and there is an ongoing effort in the definition of a role-based access control standard[15]. The central notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies management of permissions. Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles are needed.

The RBAC model has the following components as illustrated in Figure 3.

1. U, R, P and S (users, roles, permissions and sessions respectively), where P is the Cartesian product of operation *OP* and objects *Obj*,
2. $PA \subseteq P \times R$, a many-to-many permission to role assignment relation,
3. $UA \subseteq U \times R$, a many-to-many user to role assignment relation,
4. $user\_sessions : U \rightarrow 2^S$, a function mapping each user u to a set of sessions.
5. $session\_roles : S \rightarrow 2^R$, a function mapping each session s to a set of roles $session\_roles(s) \subseteq \{r|(user(s), r) \in UA\}$ (which can change with time) and session s has the permissions $\bigcup_{r \in session\_roles(s)} \{p|(p, r) \in PA\}$.

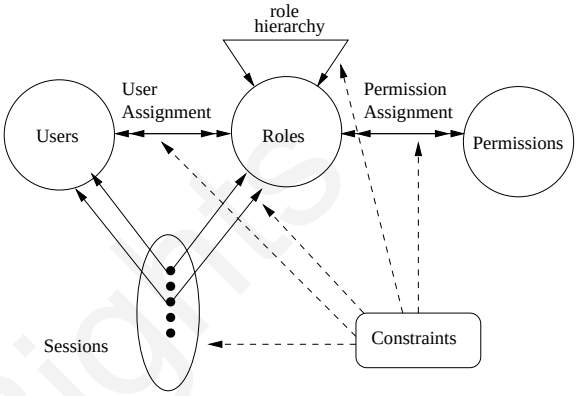

Figure 3. An RBAC model

3.2. Security Constraints Specification

With respect to RBAC, security specification constraints can apply to operations on objects, user assingment, permission assignment, and session. Constraints are predicates which, applied to these relations and functions, return a value of "acceptable" or "not acceptable". Constraints can also be viewed as sentences in some appropriate formal language.

Operation Constraints Operation constraints are specified to filter the global sensitive data. These constraints aim at handling the security policies violation generated from the collaboration among heterogeneous data sources.

In the scenario that the user is allowed to retrieve data *name* and *xray* from source s_1 and s_2 by *ssn* respectively, but the tuple *(name, xray)* is sensitive global information. When the constraints on the local sources can not protect the information properly by its own, the global level security checking need be enforced into the mediator specifications. It can be used to automatically perform global security checking on sensitive global data retrieval.

The following predicate is specified to filter the global sensitive data $(name, xray)$ by checking the mediator views: $\forall v, name_0, xray_0, ssn_1, name_1, ssn_2, xray_2$.

Copy Rights

$(\neg echo(v, name_0, xray_0) \wedge (echo(v, ssn_1, name_1) \wedge echo(v, ssn_2, xray_2) \rightarrow ssn_1 \neq ssn_2))$,
where predicate $echo(v, name_0, xray_0)$ denotes that the view v can simultaneously feedbacks the attribute $name$'s value $name_0$ and $xary$'s value $xray_0$, and $echo(v, ssn_1, name_1)$ denotes that the view v can simultaneously feedbacks the attribute ssn's value ssn_1 and $name$'s value $name_1$. Similarly does $echo(v, ssn_2, xray_2)$. This constraint is applied to the mediator operations.

User Assignment Constraints For the separation of duties principle, the same user can not be assigned to any two mutual exclusive roles simultaneously. Mutual exclusion in terms of user assignment specifies that one individual cannot be a member of both roles in exclusive sets. For instance, one user can not play *patient* and *doctor* role in the same hospital simultaneously. The following predicate prevents assignment of the mutual exclusive roles r_1 and r_2 to the same user simultaneously. $\forall u_1, u_2 \in U.(r_1 \in assigned\_role(u_1) \wedge r_2 \in assigned\_role(u_2) \rightarrow (u_1 \neq u_2))$, where $assigned\_roles(u)$ denotes the set of roles assigned to the user u.

Prerequisite role constraint may be required for the user assignment. Prerequisite roles means that a user can be assigned to role A only if the user is already a member of role B. The following predicate enforces the prerequisite role r_1 of r_2 to be assigned to user if r_2 is assigned to user. $\forall u.r_2 \in assigned\_role(u) \rightarrow r_1 \in assigned\_role(u)$. These constraints are applied to the user assignment.

Permission Assignment Constraints *Mutual exclusive permissions* constraint means that the same permission can be assigned to at most one role in a mutually exclusive set. The following predicate denotes that the mutually exclusive roles r_1 and r_2 can not be assigned same permission: $\forall p_1, p_2, p_1 \in assigned\_permission(r_1) \wedge p_2 assigned\_permission(r_2) \rightarrow p_1 \neq p_2$.

Prerequisite permissions constraint means that a permission p_1 can be assigned to a role only if that role already possesses the prerequisite permission p_2. $\forall r.p_1 \in assigned\_permission(r) \rightarrow p_2 \in assigned\_permission(r)$.

Session Constraints Constraints can also apply to sessions, and the *user* and *roles* functions associated with a session. It may be acceptable for a user to be a member of two roles but the user can not be active in both roles at the same time. The following predicate denotes r_1 and r_2 can not be the active roles for the same user simultaneously: $\forall u_1, u_2.r_1 \in active\_role(u_1) \wedge r_2 \in active\_role(u_2) \rightarrow u_1 \neq u_2$.

A role hierarchy can be considered as a constraint. The constraint is that a permission assigned to a junior role must also be assigned to all senior roles. The goal of the RBAC specification is to enforce the security specification to the mediator specification.

4. Security Enforcement

4.1. Security System Architecture

The security enforcement architecture (Figure 4) is introduced from conceptual security mediation architecture in Section 2. In this section we discuss in detail each component of the architecture and its role in disseminating secure information.

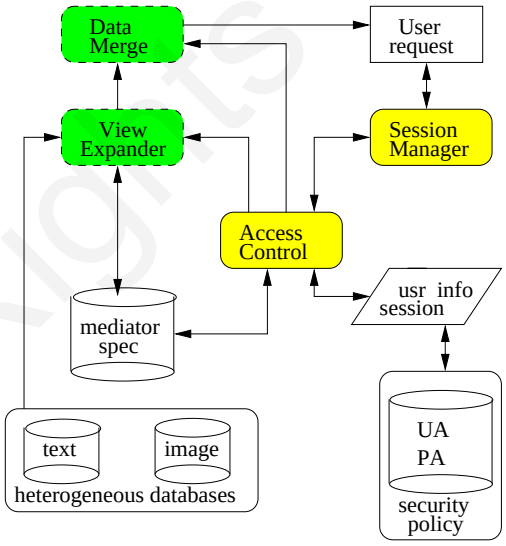

Figure 4. Security enforcement architecture

The access control module is the architecture's key component because it interfaces with other functional modules and information repositories to extract relevant information while making authorization decisions. The access control module extracts the policy information (security specification) from the policy base and works closely with the session manager module and the view expander module to enforce authorization constraints. The session manager module captures user credentials and monitors session activities to affect access control decisions. The view expander module gets information from access control module about the access permissions allowed on the view associated with an access request and generates the view rewriting plan. The result data extracted from the data sources is merged in the data merge module. The mediator specification includes the

Copy Rights

capability of data sources and mediator specified by the datalog. User requests are received as logic query and the answers are returned as instantiated view respectively.

4.2. Enforcement Method

We apply the aspect-oriented approach to enforce mediation security system [18]. The base model is specified by the mediator component specification and the aspect model is specified by the RBAC-based security aspect specification. Aspect weaving is to process the components specification and the aspects specification, and to compose them properly into an integrated specification. Essential to aspect weaving is to specify the join points, where the functionality components and security aspects interact, and to define advices, which encode the appropriate behaviors at the join points.

The component specification language is datalog and the join points are security-related predicates. The aspect specification laguage is based on first-order logic. Once a join point is met, we add some operations after the join point. In addition to Datalog system, a theorem prover such PVS [12] can be used to automate the query process.

5. Conclusion

This paper has presented a mediation security architecture for information integration based on role-based access control (RBAC). We propose a new security data model called AXM, which homogenizes data representation for heterogeneous sources. Security requirements of mediation systems are specified by constraints over various RBAC dimensions. An aspect-oriented method for security enforcement is proposed, which supports adaptive and scalable design of secure mediation systems. Our architecture can be applied to an enterprise-wide application that disseminates secure information.

We are investigating the granularity of the protected object as well as access control for multiple security policies. In order to allow the autonomy of the security policy specification at site-level, we will specify the security policy at both the organization-level and site-level. Policy conflict analysis and resolution will also be investigated.

References

[1] S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *SIGMOD Conference*, pages 137–148, 1996.

[2] C. Altenschmidt and J. Biskup. Explicit representation of constrained schema mapping for mediated data integration. In *2nd Workshop on Databases in Networked Information Systems (DNIS2002)*, pages 103–132, Aizu, Japan, 2002. Lecture Notes in Computer Science 2544, Springer, Berlin.

[3] J. Biskup and D. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Source Information Systems archive*, 28(3):169–212, May 2003.

[4] M. Carey and L. H. et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *International Workshop on Research Issues in Data Engineering: Distributed Object Management*, Taipei, 1995.

[5] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *The Eleventh International WWW Conference*, Hawaii, US, 2002.

[6] R. K. Ege, L. Yang, Q. Kharma, and X. Ni. Three-layered mediator architecture based on DHT (in press). In *International Symposium on Parallel Architcture, Algorithm and Networks (I-SPAN)*, Hong Kong, 2004. IEEE press.

[7] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[8] A. Y. Halevy. Theory of answering queries using views. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):40–47, 2000.

[9] Q. Kharma and R. K.Ege. Efficient searching via DHT in a mediator-based multimedia delivery framework (submitted). In *SPECTS2004*, 2004.

[10] D. Liu, K. Law, and G. Wiederhold. CHAOS: An active security mediation system. In *Conference on Advanced Information Systems Engineering*, pages 232–246, 2000.

[11] J. McLean. Security models. In J. Marciniak, editor, *Encyclopedia of Software Engineering*. Wiley & Sons, 1994.

[12] S. Owre, N. Shankar, J. Rushby, and D. Stringer-Calvert. *PVS Prover Guide*. Computer Science Laboratory, SRI International, Menlo Park, CA, September 1998.

[13] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proc. ICDE Conf.*, pages 251–60, 1995.

[14] R. Sandhu and E. Coyne. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[15] R. Sandhu, D. Ferraiolo, and R. Kuhn. The NIST model for role-based access control: Towards a unified standard. pages 47–64.

[16] V. Vassalos and Y. Papakonstantinou. Expressive capabilities description languages and query rewriting algorithms. *Journal of Logic Programming*, 43(1):75–122, 2000.

[17] G. Wiederhold. Mediators in architecture of future information systems. *IEEE Computer*, 25:38–49, 1992.

[18] L. Yang and R. K. Ege. Modeling and verification of real-time mediation systems. In *Advanced Simulation Technologies Conference (ASTC04)*, pages 61–68, Arlington, Virginia, April 2004.

[19] L. Yang, R. K. Ege, and H. Yu. Enhancing mediation security by aspect-oriented approach. In *Softerware Engineering and Knowledge Engineering (SEKE04)*, Banff, Alberta, Canada, June 2004.

[20] B. Zhao. Collaborative access control. In *Seminar on Network Security 2001*. Telecommunications Software and Multimedia, 2001.

Copy Rights