CHAPTER 7
SECURITY ARCHITECTURE OF THE MULTIMEDIA MEDIATOR

Christian Altenschmidt
Joachim Biskup
Yücel Karabulut
Fachbereich Informatik
Universitiät Dortmund
D-44221 Dortmund
Germany
altensch, biskup, karabulu@ls6.cs.uni-dortmund.de

Abstract Mediation is a powerful paradigm for advanced interoperable information systems. This paper presents the security module of the multimedia mediator which enforces a previously reported approach to secure mediation. In this approach, a user submits cryptographically signed credentials containing both personal authorization attributes and his public encryption key, and data sources decide on the query access on the basis of shown personal authorization attributes and return encrypted answers. The security module uniformly represents the query access authorizations of the sources, controls the intermediate usage of credentials, assists users in submitting appropriate credentials, selects and forwards credentials for subqueries, and exploits credentials for query optimization.

Keywords: Security, Mediator, Credential, Authorization, Access Control

1. INTRODUCTION

Mediation is a powerful paradigm for advanced interoperable information systems [Wie92]. A *mediator* manages queries on behalf of a user by identifying and addressing appropriate subqueries to heterogeneous and autonomous data sources and by subsequently collecting and integrating the returned answers. A previously reported approach to *secure mediation* [BFK99, BFK98] is based on a public-key infrastructure and cryptographically signed *credentials* that encode the eligibility of users. These technologies potentially provide for an inherently scalable and secure mechanism for widely distributing assured authentication and authorization attributes. Secure mediation is roughly outlined as follows [BFK99, BFK98]. A user submits evidence of being eligible for seeing the answer to a query by submitting certified *personal authorization attributes* which are encoded in credentials. A mediator examines, selects and forwards submitted credentials together with appropriate subqueries to the data sources. A data source autonomously bases *access decisons* for requested data on shown credentials, and it returns subquery answers in *encrypted form* to the mediator. For encryption it applies the user's *public key* for an asymmetric encryption scheme which is also contained in the credentials. Then the mediator processes the returned encrypted data in order to produce the final, still encrypted query answer.

In our Multimedia Mediator, MMM, project [BFKS97] a specific kind of a mediator has been designed and implemented as a prototype. A *security module* for the MMM is being constructed under the following requirements: (1) implement the design [BFK99] as outlined above, (2) smoothly integrate the security features into the functionalities of the MMM prototype, (3) meet emerging standards [IET00, RL98, PKI00] for credentials.

In this paper, we present and discuss the architecture of this security module emphasizing the following original contributions:

- an *authorization model* that allows to consider *credentials* as grantees, to be used for representing the query access authorizations of the sources;
- specifications of query access authorizations as *annotated ODL declarations*, which can be communicated from the sources to the MMM, in particular;
- a *schema authorization policy for mediation*, which allows to dynamically generate external schemas for spontaneous users;
- an *instance authorization policy for mediation*, which conjunctively combines the access restrictions of the mediator and the sources;
- an *isolated co-existence* of the functional layers and the security layers treating authorization data in close correspondance to functional data;
- an enrichment of (so far only functional) *embeddings* [ABFS98] from an application object into the proxy objects for source items, as maintained by the MMM, for evaluating expected access decisions of the sources;
- user support for *credential management* by determining implications among sets of personal authorization attributes;
- and exploitation of credentials for *query optimization.*

2. ENVIRONMENT OF SECURE MEDIATION

The Multimedia Mediator prototype. The Multimedia Mediator, MMM, is implemented as an autonomously operating *agent.* It is based on the object-oriented database management system O2. Figure 2 shows its environment and its functional and security architecture. Seen from the user, the MMM appears as an object-oriented database which consists of an *application schema* with a mostly virtual *instance.* Basically, in addition to persistent data owned by the MMM, this instance is only transiently and partially materialized by *proxy objects* representing items of the data sources. A user addresses the MMM by means of his personal *user agent.* And data sources are connected to the MMM with the help of *wrapper agents.* Interoperability is supported by employing CORBA [OMG98] for data exchanges, KQML [FLM97] for agent communications, and ODL/OQL [CB00] for schema declarations and query expressions.

Authorization model. Credentials [IET00, RL98, Cha85, PKI00, BFL96] are powerful means to receive and to present assured digital certificates about a wide range of personal properties. Such properties may include the ownership of a public key (and the corresponding secret key) for an asymmetric cryptographic scheme, a unique identification for authentication, personal properties like date of birth, gender, educational degrees, profession and so on, and received or payed permissions to get access to digital services. Moreover credentials can be used to approve that several of such properties belong together.

For mediated information systems we employ credentials that contain a *public key* and additional properties of any kind as follows: they approve that the owner of the public key enjoys the additional attributes. When shown to a data source, as well as to a mediator, the additional attributes are interpreted as *personal authorization attributes* proving that the owner of the private key is eligible for querying certain data. Thus for our purpose we abstract *credentials* as pairs of the form [*public (encryption) key, set of personal authorization attributes*] which are digitally signed by the issuer.

Any agent autonomously decides on the specific interpretation, i.e., which data it is willing to return. This means in terms of the usual authorization models that an *authorization subject* can be modelled as a set of personal authorization attributes, like identified users or roles in more traditional approaches. Accordingly, any agent needs two features to decide on an actual permission of a requested query:

- An *authorization database* of granted query access authorizations that have sets of personal authorization attributes as grantees, and
- *authorization policies* to evaluate the request with respect to the authorization database.

Figure 1 shows a coarse model for the *authorization database.* As usual, an *authorization* specifies a grantor, a privilege and a grantee. A *privilege* is composed of an *authorization object* and a set of applicable access *methods.* The *grantor* is supposed to be an identified *user* that in most cases is the *owner* of the privilege (here usually the administrators of the MMM and of the

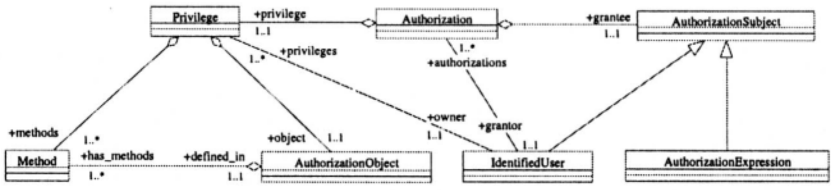

Figure 1. Coarse model for the authorization database

sources, respectively). The *grantee* might be an identified user, as usual, but additionally, as a particularity of our credential based access control, a *Boolean expression over personal authorization attributes* (*authorization expression* for short). For the sake of simple exposition, we only consider authorization expressions over atomic personal authorization attributes which are in *disjunctive normal form without negations.* Such an authorization expression can be identified with a set (representing the disjunction) which contains sets (representing the conjunctions) of personal authorization attributes as elements.

A particular instance (p, g, a) with privilege p, grantor g, and authorization expression a as grantee means that *g has granted* privilege *p* to any request which is accompanied with an authorization expression r such that "*r qualifiesfor a*". In the full model, the exact definition of the relationship "*qualifies*" formalizes the rough intuition of "*logical implies* under additional assumptions".

Then any *authorization policy* evaluates a request *on the basis* of the valid *qualifications* between the accompanying authorization expression and the required authorization expression, for all needed privileges. There is a special authorization policy, called *pure*, that *only* checks the envolved qualifications, i.e., the request is permitted iff all these qualifications are valid (and there are no further conditions).

Now we make a subtle distinction for the MMM:

- For each connected source, the MMM maintains a *representation* of the source's authorization database, and the MMM *expects* that the source applies the *pure* authorization policy.
- The mediator itself applies two more elaborated *authorization policies for mediation*, one for querying *schemas* and another one for querying *instances*, which combine the *envolved qualifications* with *additional considerations*, to be presented in subsequent sections.

Annotated ODL declarations. We mostly follow the ODMG proposal to achieve interoperability among our agents with respect to schema declarations. Thus all schema declarations are supposed to be convertable into a canonical form expressed in ODL. For the current prototype we use the following basic ODL features: *class*; *class attribute*; *attribute type* built from classes and atomic types (`boolean, string, integer,...`) using constructors like `set_of, list, struct (tuple_of)`; *relationship* (specifying inverse attributes); *key*; *extent* for denoting an access path to a full class extension; *persistent root* for denoting an access path to an arbitrary subset of a class extension. (Strictly speaking, "persistent root" is an O2 feature which is not explicitly mentioned in [CB00].) Further advanced features (*method, inheritance, interface, exception,...*) could be added to future versions of the prototype.

For specifying access authorizations in the context of the MMM, we have to identify the possibly relevant instances (p, g, a), with privilege $p = (o, m)$ for accessing an authorization object o by method m, grantor g, and authorization expression a. For the current prototype, as well as for the sake of short presentation, we make the following simplifying assumptions: (1) We restrict to *schema based* authorizations (refraining from content based authorizations which depend on actual data values). Thus as *authorization objects* we allow concrete incarnations of all ODL features, except for types where only components of struct-declarations are considered. (2) We assume some generic *access* method for all authorization objects (refraining from specializing the generic method into *read, navigate,* and so on). (3) As a default, we postulate an administrator who acts as *owner* of any object and as *grantor* of any authorization. Thus we can ignore these components further on. (4) In summary, we can specify authorizations as (o, a) where o is a considered ODL feature (more precisely a concrete incarnation of it) and a is an authorization expression. Giving these assumptions, we can succinctly express access authorization by adding two kinds of annotations to ODL schema declarations, each of which consists of an authorization expression:

- a *schema access annotation* grants a privilege to access some part of the schema itself, and
- an *instance access annotation* grants a privilege to access some part of the (mostly virtual) instance.

More precisely we offer the following possibilities: A *schema access annotation* on any ODL feature grants the privilege to access (read and use) the declaration of this very instance. An *instance access annotation* on a class c grants the privilege to access (activate) any instance object of the class extension. An *instance access annotation* on a class attribute *attr* grants the privilege to access (execute) the attribute for any of the class's instance objects that already has been accessed before. Depending on the type of the attribute, either the stored (complex) value is returned or the stored object identifier is dereferenced resulting in an access to the identified object. An *instance access annotation* on a relationship *rel* is treated as if *rel* was an attribute. An *instance* *access annotation* on a component *com* of a struct-declaration of an attribute type grants the privilege to access that component of an instance object provided the struct part has been accessed before. An *instance access annotation* on an extent or persistent root *ext* grants the privilege to access that data structure, i.e., to execute all available set operations including scanning, searching and so on.

At this point we emphasize that so far we have only described how an annotation updates the *authorization database*. In section 3 we define the *authorization policies* which finally regulate the actual access decisions.

3. SECURE MEDIATION

For secure mediation we distinguish the initialization phase and, afterwards, preparatory phases for sources and users, respectively, and query request phases and corresponding answer delivery phases.

Initialization phase. In the *initialization* phase, a mediation administrator *instantiates* the MMM by declaring an *application schema* expressed in ODL which captures the information needs of the anticipated users. For many parts of the application schema, there will be no persistent instance in general but the corresponding data is dynamically retrieved from sources at query time.

But for some parts of the schema, the administrator may want to maintain data owned by the MMM itself. Technically, for such parts the administrator additionally declares a *twin schema* as a subschema of the application schema, and he inserts an instance for this twin schema. Later on, query processing treats the twin schema and its instance, also referred to as *twin source*, nearly like the other sources.

While specifying the application schema and the twin schema the administrator also grants the (*query access*) *authorizations* using annotated ODL declarations as introduced in section 2 for both schemas. If there is no annotation for a concept, then there will be no query access restrictions. Conceptually, the authorizations for both the application schema and the twin schema are stated independently, though, in practice, they usually will be closely related.

Preparatory phase for a source. In a *preparatory phase for a source*, a data source can be connected with the MMM for further cooperation provided the source can comply with the functional requirements of the instantiated MMM. Furthermore, the source must provide a credential based authorization policy. The basic functional requirements are that appropriate parts of the application schema can be *embedded* in matching parts of the (virtual or existing) source schema. If the requirements are met, an appropriate *wrapper* agent is created which is devoted to convert source items into a form that is canonical for the mediator, and vice versa.

Thus, in this phase, firstly the wrapper provides a canonical form of the matching source schema parts, again expressed in ODL, which subsequently are internally represented within the MMM and there connected to the matching application schema parts by so-called embeddings [ABFS98]. Secondly, as far as possible, the wrapper also converts the source's authorization database into a canonical form. For this purpose, the wrapper adds pertinent annotations to the ODL representation of the source schema, which are subsequently internally represented within the MMM.

Preparatory phase for a user. In a *preparatory phase for a user*, a spontaneously emerging user asks for inspecting the application schema of the MMM because he wants to know how it captures his information needs. In order to do so, he presents some of his credentials. Let us assume that these credentials contain the set of personal authorization attributes which is denoted by the authorization expression *a*. Then the MMM returns the largest *subschema* of the application schema permitted to that user according to the following *schema authorization policy for mediation*:

- A subschema is *permitted* (for an authorization expression *a*) iff it is allowed according to the pure authorization policy applied to the authorization database generated from the annotations declared for the application schema (so far the privileges for all subschema items are granted individually) and, additionally, 1.) the subschema is syntactically correct (closed with respect to the ODL rules) and 2.) privileges are consistently granted (closed with respect to inferences in the sense of [OvS94, CG98]).

Thus the user gets a dynamically generated *external view* about the *functionality* of the mediator. This view is *closed* in the following sense: 1.) If an item is shown, then all other items needed to access that item are also shown. 2.) It is impossible to infer the existence of further items that are not shown. The *dynamic* generation of subschemas is in the spirit of the open computing environments for mediation. Hence we favour this novel dynamic approach rather than declaring a limited set of external views in advance.

On demand, the user also gets the *authorization requirements* for the corresponding (virtual) instances, i.e., the user can ask to be informed about which personal authorization attributes are *likely to be sufficient* to access which parts of the corresponding (virtual) instance. Then the user can assemble credentials with his personal authorization attributes *potentially* providing evidence of his eligibility to see answers to submitted queries.

Query request phase. In a *query request phase,* a prepared user sends a global *request* to the MMM. A global request includes a global query with respect to the application schema and a set of credentials. The MMM analyzes the request with respect to functional and authorization requirements, thereby identifying subrequests to be forwarded to connected sources. Hereby, the twin source maintained by the MMM itself is treated like any external source, except that no wrapper is involved.

Accordingly, a *subrequest* to a source includes a local query with respect to the internal representation of the source schema and appropriately selected credentials, which are subsequently converted by the responsible wrapper.

Concerning the authorization requirements, the MMM relates the global authorizations of the MMM to the local authorizations of the connected sources using the following *instance authorization policy for mediation*:

- A request is (globally) *permitted* (for an authorization expression *a*) iff it is allowed according to the *pure* authorization policy applied to the authorization database generated from the annotations declared for the application schema and, additionally, all subrequests are (locally) permitted.
- A subrequest to the twin source is (locally) permitted iff it is allowed according to the *pure* authorization policy applied to the authorization database generated from the annotations declared for the twin schema.
- A subrequest to an external data source is (locally) permitted iff the MMM *expects* that the source will permit the access as requested by the included local query based on the included credentials. The *expectation* is based on the *pure* authorization policy applied to the authorization database generated from the annotations specified for the representation of the source schema.
- If a request is (globally) permitted, then all subrequests are forwarded and then autonomously evaluated by the sources. Otherwise the request is (globally) *denied*, and the user will be informed that no answer can be returned due to the lack of personal authorization attributes.
- An external data source, as well as the twin source, autonomously decides on permitting subrequests.
- Subanswers from the twin source and external sources are processed in the mediator and forwarded to the user without any further restrictions.

Thus, seen from the user, the authorization requirements of the mediator and the sources are *conjunctively* combined: access to source data via the mediator is permitted iff it is allowed by *both* the mediator *and* the source. Accordingly, the security module of the mediator acts as a kind of *filter* put in front of the sources. There are obvious tradeoffs between the strength of the filters as defined by the global authorizations, the overall run-time complexity of mediated query evalution and the response quality. These tradeoffs can be exploited for query optimization as discussed in the preproceedings.

Answer delivery phase. A query request phase is followed by the corresponding *answer delivery phase* (allowing the interleaving of several requests). In this phase, a source produces a *protected subanswer*:

- All *content data* for attribute values is piecewise *probabilistically encrypted* with the public encryption key of the user which is connected with those personal authorization attributes on the basis of which query access permissions have been decided.
- All *structural parts* are left *open as plain text.*

The MMM uses the protected subanswers to generate new proxy objects, and if necessary new application pseudo objects which are associated with the corresponding proxy objects. Pertinent embeddings are determined by the types of the application pseudo objects and the types of their corresponding proxy objects in order to define (usually encrypted) attribute values for old and new application pseudo objects. As soon as all subanswers are available (or after some timeout) a suitably modified version of the original (global) query is evaluated on the basis of the current instance, the embeddings into the proxy objects and the retrieved method values for the proxy objects.

Surely, the functionality of this phase is essentially restricted by the attribute values being encrypted with public keys of the user. Basically, the full functionality is preserved as far as the modified version of the original query does not have to perform comparisons for selections and joins on encrypted values. For the strategies to avoid the restrictions we refer to the preproceedings.

4. THE SECURITY MODULE

In this section we provide some more details on how *secure mediation* as explained in section 3 is actually implemented as part of the MMM. Figure 2 gives a first rough overview about the data flow between functional and security components. We only describe the main security components. Further details are given in the preproceedings.

The security module of the MMM has three main components, the security knowledge base, SECKNOB, the credential manager, CREMA, and the mediator authorization decision engine, MAIDEN.

The security knowledge base, SECKNOB, maintains the following parts: (1) The *authorization databases* that are generated from the annotated declarations of the application schema, the twin schema and the representations of the external source schemas. (2) The *credentials* submitted by a user. (3) The *authorization attributes* extracted from the credentials together with links associating them with the credentials they are extracted from. (4) A (Horn clause) *rule base* that specifies implications among authorization expressions which are invoked when an instance of the relationship "*qualifiesfor*" between

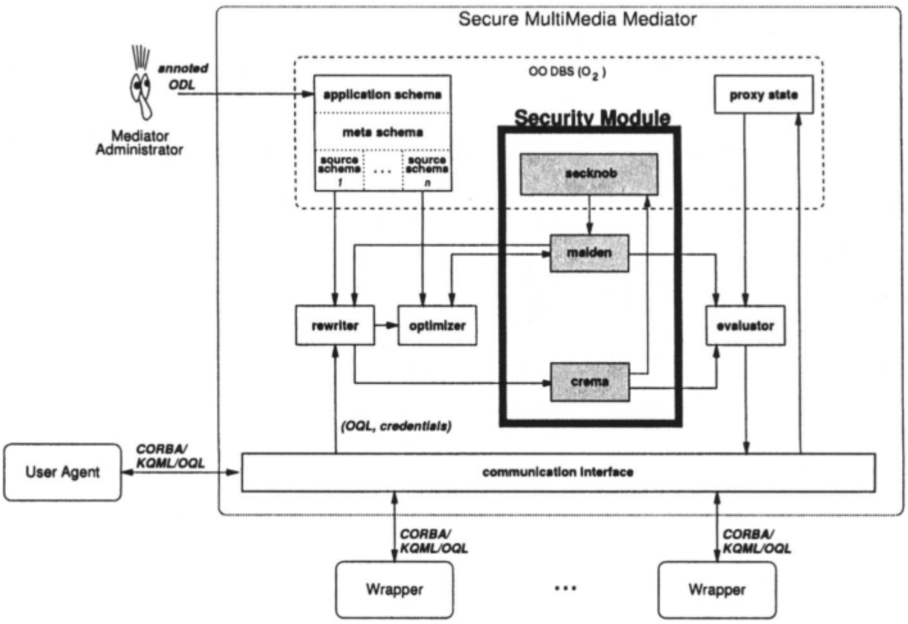

Figure 2. Security architecture of the Multimedia Mediator

authorization expressions must be decided. The rule base is intended to take advantage of "real world" properties of authorization attributes, and to encode known access control features like structured objects, groups or roles, for each of which we can define hierarchies and corresponding inheritance rules. (5) A collection of *protocol converters* which unify the various formats proposed in existing and emerging standards for credentials [IET00, RL98, PKI00].

The credential manager, CREMA, verifies the authenticity and validity of submitted credentials by checking the approving digital signatures and expiration dates. Moreover, it extracts authorization attributes from credentials, inserts credentials, their authorization attributes and the pertinent links into SECKNOB, and determines credentials to be included to subqueries.

The mediator authorization decision engine, MAIDEN, implements the *authorization policies for mediation*, both for *schemas* and for *instances*, and the underlying *pure* authorization policy together with its basic relationship "*qualifies for*" among authorization expressions.

Additionally, we need an appropriate access control shell for the underlying object-oriented database management system. Unfortunately, this shell is not provided by O2 which we use for our prototype.

5. RELATED WORK AND CONCLUSION

Security in interoperable information systems has mostly been investigated within federated database contexts, where the emphasis laid on resolving heterogeneity [Jon98, TF97]. For contributions to security in mediated systems see [CJS96, WBD98, DSS00]. The security mechanisms presented in the works above are identity based rather than credential based.

With our credential based approach we can model both *identity based authorization* as well as *attribute based authorization.* In contrast to the efforts above, our approach makes a specific contribution towards interoperation by combining the credential based authentic authorization with some kind of anonymity and of asymmetric encryption for confidentiality. The concept of credentials has also been adopted previously for various purposes in interoperable systems [SWW97].

The ongoing implementation is based on several original contributions, in particular the identification of suitable schema and instance *authorization policies for mediation*, ODL declarations with schema access and instance *access annotations*, and an analysis of the impact of authorization for *query optimization.* There are a lot of issues for further research and development. Among them are the exploitation of the full scope of ODL, rapid construction of "authorization wrappers", refined optimization, more sophisticated treatment of encrypted subanswers, and user support for presenting appropriate credentials.

References

[ABFS98] C. Altenschmidt, J. Biskup, J. Freitag, and B. Sprick. Weakly constraining multimedia types based on a type embedding ordering. In *Proceedings of the 4th International Workshop on Multimedia Information Systems*, pages 121–129, Istanbul, Turkey, September 1998.

[BFK98] J. Biskup, U. Flegel, and Y. Karabulut. Towards secure mediation. In *1st Workshop on Sicherheit und Electronic Commerce*, pages 93–106, Essen, Germany, October 1998.Vieweg-Verlag.

[BFK99] J. Biskup, U. Flegel, and Y. Karabulut. Secure Mediation: Requirements and Design. In *Proceedings of the 12th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 127–140, Chalkidiki, Greece, 1999. Kluwer Academic Press.

[BFKS97] J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. Query evaluation in an object-oriented multimedia mediator. In *Proceedings of the 4th International Conference on Object-Oriented Information Systems*, pages 31–43, Brisbane, Australia, November 1997. Springer Verlag.

[BFL96] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *17th IEEE Symposium on Security and Privacy*, pages 164–173, Los Alamitos, 1996.

[CB00] R. G. G. Cattell and Douglas Barry, editors. *The Object Data Standard: ODMG 3.0.* Morgan Kaufmann, San Francisco, 2000.

[CG98] F. Cuppens and A. Gabillon. Rules for designing multilevel object-oriented databases. In Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, and Dieter Gollmann, editors, *Proceedings of the 5th European Symposium on Research in Computer Security (ESORICS'98)*, number 1485 in LNCS, pages 159–174, Louvain-la-Neuve, Belgium, September 1998. Springer-Verlag.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications ofthe ACM*, 28(10): 1030–1044, October 1985.

[CJS96] K. S. Candan, Sushil Jajodia, and V. S. Subrahmanian. Secure mediated databases. In Stanley Y. W. Su, editor, *12th International Conference on Data Eng.,* pages 28–37, New Orleans, Louisiana, USA, Feb. - Mar. 1996. IEEE, IEEE Computer Society Press.

[DSS00] S. Dawson, Qian S., and P. Samarati. Providing security and interoperation of heterogeneous systems. *Distributed and Parallel Databases*, 8(1):119–145, January 2000.

[FLM97] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In J. M. Bradshaw, editor, *Software Agents.* MIT Press, Cambridge, 1997. `http://www.cs.umbc.edu/kqml/papers/`.

[IET00] IETF SPKI Working Group. SPKI certificate documentation. `http://world.std.com/~cme/html/spki.html`, July 2000.

[Jon98] D. Jonscher. *Access Control in Object-Oriented Federated Database Systems.* PhD thesis, University of Zurich, Department of Computer Science, Zurich, May 1998. DISDBIS 49, Infix-Verlag.

[OMG98] Object Management Group. The common object request broker, architecture and specification. CORBA 2.3.1/IIOP specification, `http://www.omg.org/library/c2indx.html`, December 1998.

[OvS94] M.S. Olivier and S.H. von Solms. A taxonomy for secure object-oriented databases. *ACM Transactions on Database Systems*, 19(1):3–46, 1994.

[PKI00] PKIX Working Group. An internet attribute certificate profile for authorization. Internet draft, work in progress. `http://www.ietf.org/internet-drafts/draft-ietf-pkix-ac509prof-03.txt`, May 2000.

[RL98] R. L. Rivest and B. Lampson. A simple distributed security infrastructure (SDSI). `http://theory.lcs.mit.edu/~cis/sdsi.html`, 1998.

[SWW97] K. E. Seamons, W. Winsborough, and M. Winslett. Internet credential acceptance policies. In *Proceedings of the Workshop on Logic Programmingfor Internet Applications*, Leuven, Belgium, July 1997.

[TF97] Z. Tari and G. Fernandez. Security enforcement in the DOK federated database system. In P. Samarati and R. S. Sandhu, editors, *Database Security, X: Status and Prospects, Proceedings ofthe 10th IFIP WG 11.3 Working Conference on Database Security*, pages 23–42, Como, Italy, 1997. Chapman & Hall.

[WBD98] Gio Wiederhold, Michel Bilello, and Chris Donahue. Web implementation of a security mediator for medical databases. In T. Y. Lin and Shelly Qian, editors, *Database Security, XI: Status and Prospects, Proceedings of the 11th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 60–72, Lake Tahoe, California, 1998. IFIP, Chapman & Hall.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992 .