Secure Mediation Between Strangers in Cyberspace

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik

von

Yücel Karabulut

Dortmund
2002

Tag der mündlichen Prüfung: 26. September 2002

Dekan: Prof. Dr. Bernhard Steffen

1. Gutachter: Prof. Dr. Joachim Biskup

2. Gutachter: Prof. Dr. Heiko Krumm

Abstract

Friendship is constant in all other things

Save in the office and affairs of love:

Therefore all hearts in love use their own tongues;

Let every eye negotiate for itself

And trust no agent.

- W. Shakespeare. Much Ado about Nothing.

This thesis is concerned with solutions to challenges associated with secure mediation between strangers in cyberspace. In mediated information systems clients and information sources are brought together by mediators. The mediation paradigm needs powerful and expressive security mechanisms considering the dynamics and conflicting interests of mediation participants. In this thesis we present a security framework for mediation with an emphasis on confidentiality and authenticity. We argue for basing the enforcement of confidentiality and authenticity on certified characterizing properties, such as personal authorization attributes, rather than on identification. In our security framework specification and enforcement of permissions are based on the public-key infrastructures which allow the binding of characterizing properties to public keys.

In Part I, we give an introduction and overview to technological background and fundamentals.

In Part II, we discuss the basic needs for mediation and outline two kinds of mediator approaches, called information integrating mediator and entity finding mediator. We also exhibit the core security requirements and concepts needed for the corresponding mediation approaches, called i-mediation and f-mediation, respectively.

In Part III, we present the design of a hybrid PKI model for a public-key infrastructure which is to be used for specifying and enforcing permissions in distributed

computing systems. We show how we extend and integrate previous public-key infrastructure approaches X.509 and SPKI/SDSI into a hybrid model. We also demonstrate two applications of the hybrid PKI model.

In Part IV, we first discuss the requirements on security for a specific approach to i-mediation with an emphasis on confidentiality and authenticity, we then propose a general design of secure i-mediation, and finally, we refine the general design for a specific approach to i-mediation and present the security architecture for the prototype of our Multimedia Mediator.

In Part V, we present the implementations of three interrelated demonstrators. First, we show the main features and components of an agent PKI framework. Second, we present the implementation of an agent-oriented and KQML-based hybrid PKI demonstrator, and third, we provide more details on how secure i-mediation, as discussed in Part IV, is actually implemented.

In Part VI, existing approaches for secure i-mediation and certificate/credential-based access control are discussed, classified and evaluated. We conclude by discussing the results and possible extensions of this thesis.

Acknowledgements

First and foremost, I must thank Joachim Biskup, my advisor, for his guidance, patience, and encouragement over these years. If I become a better writer because of my time as a PhD student, it is largely because of him. Without the encouragement of Joachim Biskup, writing this thesis would not have been possible.

I am especially grateful to Christian Altenschmidt and Ulrich Flegel for many productive discussions and for their contributions to the publications that form a part of this thesis.

Special thanks to bright master's students Carsten Kröger, Patrick Lehmann and Peter Hildebrand for indirectly contributing to my thinking and thereby this thesis. I benefited from many discussions with them during the time I supervised their theses.

Many thanks to Biswanath Panda, who helped in the last stages of the implementation.

I would also like to thank to Prof. Dr. Norbert Fuhr, the advisor of my master's thesis, on whose recommendation I started working with Joachim.

I would like to express my gratitude towards my doctorate committee, namely Prof. Dr. Joachim Biskup, Prof. Dr. Heiko Krumm, Prof. Dr. Horst Wedde and Dr. Peter Herrmann.

Loving thanks to my uncle Ali Inan, my cousins Kenan Akyol and Zafer Acer, and my friends Nurcan Demirbüken, Cemal Törün and Gülhan Yilmaz, who are always emotionally supportive.

But my warmest "teşekkür" goes to my parents, my sister and my brother for their unconditional support and love at all times.

This thesis is dedicated to my parents, my sister and my brother.

List of Abbreviations

ACL	Access Control List
ACREMA	Agent Credential Manager
API	Application Programming Interface
APROMA	Agent Property Manager
ASECKNOB	Agent Security Knowledge Base
ATRUMA	Agent Trust Manager
CORBA	Common Object Request Broker Architecture
CPS	Certificate Practice Statement
CREMA	Credential Manager
CRENG	Credential Engine
DAC	Discretionary Access Control
DSA	Digital Signature Algorithm
***F*-Mediator**	Entity Finding Mediator
KQML	Knowledge Query and Manipulation Language
***I*-Mediator**	Information Integrating Mediator
MAC	Mandatory Access Control
MAIDEN	Mediator Authorization Decision Engine
MMM	Multimedia Mediator

ODL Object Definition Language

ODMG Object Data Management Group

OMG Object Management Group

OQL Object Query Language

PEM Internet Privacy Enhanced Mail

PGP Pretty Good Privacy

PKI Public-Key Infrastructure

RBAC Role-Based Access Control

SDSI Simple Distributed Security Infrastructure

SECKNOB Security Knowledge Base

SHA-1 Secure Hash Algorithm

SPKI Simple Public Key Infrastructure

UML Unified Modelling Language

XML Extensible Markup Language

XSLT Extensible Stylesheet Language Transformations

Contents

List of Tables

List of Figures

Part I.

Introduction and Technical Background

1. Introduction

1.1. Background and Motivation

Recent trends in information technologies have led to vastly improved communication facilities such as the Internet, an explosion of on-line information providers, and spontaneously emerging users. In the pervasive computing environments emerging from these trends, most interactions including business transactions occur between strangers, due to billions of spontaneous users and the fact that most of them do not share a common security domain. Complete strangers are willing to enter new business relationships and to conduct secure business transactions electronically, with a trust that comes close to traditional face-to-face business interactions.

More specifically, spontaneous users wish to find information and several heterogeneous and autonomous sources wish to share their resources and aim at supporting potential clients. Naturally, sources and clients will want to interact with each other in a confidential and privacy-preserving manner. Sources usually have to protect their information with respect to confidentiality, i.e., to take care to disclose information only to those clients who are entitled to see it. Clients may need to automatically verify that sources are certified to adhere to standardized, well-known privacy practices and procedures or that sources possess evidences proving that the sources are trustworthy and eligible.

Further on, while requesting accesses to the resources, clients may be unwilling to reveal their identities for private reasons and thus prefer to remain anonymous. Additionally for a resource owner, it may be necessary to see evidences of a client's eligibility rather than to know *who* they are. According to these trends, a client proves her eligibility to see a piece of information by a collection of her characterizing properties, such as personal authorization attributes (e.g. profession, organizational membership, security clearance, academic title), and each autonomous source follows a security policy that is expressed in terms of characterizing properties.

When the resource owner and the requesting client are unknown to one another and when resources are to be shared across administrative boundaries, the conventional authorization scheme fails. In emerging applications, in large-scale, open, distributed systems (e.g. Internet), often there is no relationship between a requestor and an authorizer prior to a request. Because the authorizer does not know the requestor (i.e. whose security domain is most probably incompatible with the security domain of the authorizer) directly, it has to use information from third parties who know the requestor better; normally, the authorizer trusts these parties only for certain things and only to certain degrees. This trust aspect makes authorization different from traditional identity-based access control.

Whenever a user looks for a piece of information, she may aim at identifying promising sources which can be quite *heterogeneous* and *autonomous*. Whatever data a source has to offer, it may aim at supporting a *wide range* of potential clients, which are in general unknown in advance and may belong to *heterogeneous* and *autonomous* security domains. In order to meet these challenging demands, additionally *mediation* is used.

In mediated information systems [91, 95], a client seeking information and various autonomous sources holding potentially useful data, are brought together by a third kind of independent components, called *mediators*. Mediation is required to deal with heterogeneity and the autonomy of the sources, not only from the functional point of view, but also with respect to all aspects of security, such as confidentiality and authenticity.

Considering the dynamics and the security requirements of mediation participants, we argue that the employment of merely identity-based authentication and authorization approaches is inadequate and ineffective for pervasive computing environments such as secure mediation. Seemingly, secure mediation requires a powerful and expressive *property-based* approach to authorization, taking the anonymity needs of the clients into account. This approach should be based on evidences of clients' eligibility rather than user authentication and access control based on user identities.

Like in a marketplace of supply and demand, there exists different motivations for cooperation between the participants in mediation. The interoperability in mediated information systems may be stimulated by different participants having different motivations and security requirements. Depending on the motivation behind a stimulation, different kinds of mediators are needed for achieving different goals. In this thesis, we consider two kinds of mediators and the corresponding mediation approaches, as outlined in the following.

Clients demand systems enabling them to effectively work with heterogeneous information sources. This demand stimulates information sources to supply their information on an ad-hoc basis, in particular for purchase. Using mediators, clients can find and correlate information more efficiently. In such scenarios, the mediation participants appear in the following roles:

- A *client* is characterized by her assigned properties and seeks for information.
- Each heterogeneous and autonomous *source* offers data and follows a security policy expressed in terms of characterizing properties.
- A *mediator* has two major functions:
    - From the functional point of view, the main role of a mediator is to retrieve, homogenize and assemble data from any sources the mediator may find worthwhile to contact.
    - From the security point of view, the mediator contacts only the sources, whose security policies match a client's characterizing properties. Thus, seen from the client, a mediator acts as a kind of *filter* put in front of the sources.

We call such a mediator an *information integrating mediator*, also called an *i-mediator*. The process of mediation using *i*-mediators is called *i-mediation*.

A second kind of mediator can be used to act as a broker between independently operating security domains of the mediation participants. Sources demand systems enabling them to determine a wide range of potential clients which could be interested in requesting specific services of the sources and which are qualified in terms of evidences of their eligibility. Information sources may supply their information for purchase as well as for collaboration.

The conceptual challenge arising in this situation concerns how remote and autonomous entities can agree on a common understanding of properties and other issues related to these properties (e.g. certificate/credential formats) which are needed to decide whether an entity is eligible with respect to a protected service offered by another entity. On the one hand, properties are assigned to clients in their autonomously operating security domains, in principle without knowing their later usage. On the other hand, sources independently define their security policies in terms of these properties. Seen from the sources' point of view, the sources wish to be assisted to determine potential eligible clients. Seen from the clients' point of view, clients wish to be assisted to assemble appropriate properties and corresponding certificate/credentials in order to receive the information services they want.

To reach potentially eligible clients, a source could use a specific mediator having the required domain expertise as well as the relationships with the potential clients. A mediator could mediate between the security policy of a source and clients' characterizing properties which have been independently defined by the resource owner and asserted by some trusted parties vouching for the clients' characterizing properties, respectively. Sources could reach a wider range of potential clients by using such mediators. A mediator of this kind has the following major functions:

- From the functional point of view, the main role of a mediator is to seek out an entity B for another entity D and stimulate B to contact D.
- From the security point of view, the mediator acts as a broker between independently operating security domains of the mediation participants by mapping the properties and the security requirements of an entity B on the properties and the security requirements of the other entity D.

We call such a mediator an *entity finding mediator*, also called a *f-mediator*. The process of mediation using f-mediators is called *f-mediation*. The key concept of the f-mediation is the delegation of authorities. In f-mediation, an entity D *delegates* the authority over finding an eligible entity B to a f-mediator.

Considering these trends, in this thesis we aim to achieve the following goals:

- Development of a flexible hybrid PKI model. This model should constitute a trust management infrastructure which is to be used not only for mediation, but also for many other applications.
- Design and implementation of an elaborated secure i-mediation approach with an emphasis on confidentiality and authenticity.

- Implementation of a hybrid PKI demonstrator in order to prove the key ideas underlying the hybrid PKI model.

- Implementation of a flexible agent PKI framework providing core PKI services needed by the implementation of security module of the prototype of our Multimedia Mediator and by the hybrid PKI demonstrator. This framework should also offer a variety of prevalent programming interfaces for providing core PKI services to many other PKI-enabled applications.

In this thesis we follow an agent-oriented and object-oriented approach. We employ CORBA for data exchanges and KQML for agent communications. We mostly follow the ODMG proposal to achieve interoperability among our agents.

1.2. Contributions

This thesis aims at developing a security framework for mediation between strangers to achieve the goals that were sketched in the preceding section. The contributions of this thesis can be grouped into four categories, beginning with the general concepts and developing towards the more specific contributions:

1. Certificate/Credential-based secure mediation: We discuss the needs for secure mediation. We classify the mediation as i-mediation and f-mediation according to different employment goals of mediators in distibuted information systems. We identify the main characteristics of a security framework supporting property-based security policies which are suitable for secure i-mediation and secure f-mediation.

Our secure mediation proposals are based on a property-based approach to authorization, taking the anonymity needs of the clients into account. Traditional access control mechanisms, as employed by existing secure federated database solutions [55, 54, 85, 28] and also secure i-mediation solutions [23, 92, 93, 94, 27], operate under a closed world assumption in which all of the entities are registered and locally known. In contrast to these works, we argue for basing the enforcement of security goals, in particular confidentiality and authenticity, on public-key infrastructures which generalize traditional mechanisms by eliminating the closed world assumption.

2. A flexible hybrid PKI model: Previous approaches for defining a PKI are classified as based either on trusted authorities with licencing (e.g. X.509) or on owners with delegations (e.g. SPKI/SDSI, KeyNote).

A trust management infrastructure, as part of the aimed security framework, for certificate/credential-based secure i-mediation and f-mediation has to use and to link both kinds of PKI. Accordingly, we present a *flexible* hybrid PKI model which unifies and extends the previous approaches. Flexibility means that the hybrid PKI model can be used in open distributed systems not only for mediation, but also for a broad spectrum of PKI-enabled applications. While developing such a hybrid PKI model, we make the following contributions:

- We present property assignment and trust evaluation, which are two basic functionalities of any PKI, in a generic and unified conceptual model. It is generic, since two prevalent PKI models can be instantiated from this conceptual model. It is unifiying, since previous approaches uses different concepts and terms, some of which, are overloaded. Our conceptual model gives a unified view on existing models.

- We introduce explicit *administrative properties* to be encoded in supporting documents.

- We introduce *bound authorization attributes* which are more general than capabilities.

- We exhibit the *generic structure of links* between instances of the previous PKI models. Basically, in such a link an entity employs its security policy in order to convert submitted certificates assuring *free properties* of the holder of a public key into *bound properties* promising the holder to permit access to some service.

- We present a comprehensive class model of entities. This model represents all entities involved in two prevalent PKI models and a hybrid PKI model.

- We demonstrate the applicability and flexibility of the hybrid model by presenting two practical application scenarios. Additionally, seen from the viewpoint of the role-based access control research efforts, we exhibit how distributed

RBAC can be implemented for dynamic coalition environments by employing our hybrid PKI model.

3. An elaborated secure *i*-mediation solution: The i-mediation paradigm needs powerful and expressive security mechanisms considering the dynamics and conflicting interests of i-mediation participants. Firstly, we discuss the security requirements for i-mediation with an emphasis on confidentiality and authenticity. We argue for basing the enforcement of these properties on certified personal authorization attributes rather than on identification. Secondly, we propose a general design of secure i-mediation where accreditation-certificates are roughly used as follows: clients show their eligibility for receiving requested information by the contained personal authorization attributes, and sources and the i-mediator guarantee confidentiality by using the contained encryption keys. Thirdly, we refine the general design for a specific approach to i-mediation and present the security architecture for the prototype of our Multimedia Mediator. More concretely, we make following contributions:

- We discuss the needs of secure i-mediation, thereby emphasizing the differences and the new challenges in comparison to the more traditional federated approach.
- We present the protocols for *secure direct querying* and *mediated query answering*, and we examine their security properties.
- We analyze the achievable security properties including support for anonymity, and we discuss the inevitable tradeoffs between security and i-mediation functionality.
- We present the security architecture for the prototype of the MMM.
- We define an *authorization model* that allows considering expressions over *personal authorization attributes* extracted from *accreditation-certificates* as grantees, to be used for representing the query access authorizations of the sources.
- We define the specification of query access authorizations as *annotated schema declarations* within the framework of ODL.

- We present a *schema authorization policy for mediation*, which allows dynamically generating external views for spontaneous users.
- We present an *instance authorization policy for mediation*, which conjunctively combines the access restrictions of the mediator and the sources.
- We present an *answer encryption policy for mediation*, which supports information flow control.

Contrary to existing secure i-mediation solutions [23, 92, 93, 94, 27] which employ conventional authorization scheme, our certificate/credential-based proposal constitutes a new solution in this research area.

4. Prototypical demonstrators: Most of the security requirements presented in this thesis were implemented in three interrelated demonstrators:

- We have developed a *flexible agent PKI framework* which provides core PKI services such as encoding, issuing, and verifying certificates and credentials. The PKI framework supports a variety of PKI-enabled applications by providing a Java API and a C++ API. The PKI framework also provides a CORBA interface for CORBA-enabled applications. Due to the variety of offered interfaces, the agent PKI framework can be used not only for mediation agents (e.g. MMM), but for many other PKI-enabled applications.

  In current PKIs, such as X.509, certificates are not human readable. It is usually ambiguous, what exactly a X.509 certificate encodes. One of the main particularities of our agent PKI framework is the employment of the widely used standard XML for representing certificates and credentials.
- In order to prove the key ideas and to implement the general design of the hybrid PKI model, a specific approach has to be taken into account. For this purpose, we have implemented an agent oriented and KQML-based *hybrid PKI demonstrator*. For this purpose, we extended the works presented in [46, 86] which propose secure-communiation related and PKI-related extensions to KQML. The new extensions to KQML are used by the agent interactions that are involved in an instance of the hybrid PKI model.

- Again, in order to implement the general design of secure *i*-mediation, we have implemented a demonstrator of the security module for the prototype of the MMM. The main particularity of this demonstrator is the achievement of an *isolated co-existence* of the functional layers and the security layers treating authorization data in close correspondence to functional data.

1.3. Cooperation and Publications

The research has been conducted by the author of this thesis, in cooperation with a group of PhD students and master's students under the cooperation and supervision of Prof. Dr. Joachim Biskup. At various times, the PhD students Christian Altenschmidt and Ulrich Flegel and the master's students Carsten Kröger, Patrick Lehmann and Peter Hildebrand worked in the secure mediation group. The author of this thesis has challenged his architectural vision into three master's theses and worked together with each master's candidate in handling the finer level conceptual and implementational problems.

Some results of this thesis have been or are to be published in the following publications which (with one exception) have been co-authored with members of the secure mediation group. Where appropriate, the original text of the publications also appear in this thesis.

P1 J. Biskup and Y. Karabulut. A Hybrid PKI Model with an Application for Secure Mediation. In *16th Annual IFIP WG 11.3 Working Conference on Data and Application Security*, Cambridge, England, July 2002. to appear.

P2 C. Altenschmidt, J. Biskup, U. Flegel and Y. Karabulut. Secure Mediation: Requirements, Design and Architecture. *Journal of Computer Security*. to appear.

P3 C. Altenschmidt, J. Biskup and Y. Karabulut. Security Architecture of the Multimedia Mediator. In *14th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 77-87, Schoorl, Holland, Aug. 2000. Kluwer Academic Press.

P4 Y. Karabulut. Credential Management for Secure Mediators. In *11th GI-Workshop on Foundations of Databases*, pages 52-56, Luisenthal, Germany, May 1999.

P5 C. Altenschmidt and Y. Karabulut. A Secure Object-Oriented Mediator: Two Challenges. In *3rd Workshop on Federated Databases*, pages 65-80, Magdeburg, Germany, Dec. 1998. Shaker-Verlag.

P6 J. Biskup, U. Flegel and Y. Karabulut. Towards Secure Mediation. In *1st Workshop on Security and Electronic Commerce*, pages 93-106, Essen, Germany, Oct. 1998. Vieweg-Verlag.

P7 J. Biskup, U. Flegel and Y. Karabulut. Secure Mediation: Requirements and Design. In *12th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 127-140, Chalkidiki, Greece, July 1998. Kluwer Academic Press.

1.4. Organization of this Thesis

This thesis consists of six parts.

Part I comprises Chapters 1 and 2. Chapter 2 provides the technological background and fundamentals.

Part II consists of Chapter 3. It gives a brief introduction to the research topic by outlining two different mediation approaches and the core security concepts needed for these approaches. Chapter 3 includes a detailed[1] discussion of issues presented in the introduction and motivation part of Chapter 1.

Part III consists of Chapter 4 and 5. Chapter 4 presents the design of a hybrid model for a PKI to be used for specifying and enforcing permissions in distributed computing systems. Chapter 5 demonstrates two applications of the hybrid PKI model.

Part IV comprises Chapter 6. In this chapter, we first discuss the requirements on security for a specific approach to i-mediation with an emphasis on confidentiality and authenticity, we then propose a general design of secure i-mediation, and finally, we refine the general design for a specific approach to i-mediation and present the security architecture for the prototype of our Multimedia Mediator.

[1]Where appropriate, for a better understanding of the research topic, a substantial part of Chapter 1 is repeated in Chapter 3.

Part V consists of Chapters 7 to 9. Chapter 7 exhibits the main features and components of an agent PKI framework. Chapter 8 presents the implementation of an agent-oriented and KQML-based hybrid PKI demonstrator. Chapter 9 provides more details on how secure *i*-mediation, as discussed in Part IV, is actually implemented.

Part VI comprises Chapter 10 and 11. In Chapter 10, existing solutions for secure mediation and certificate/credential-based access control are discussed, classified and evaluated against the requirements given in that chapter. Finally, Chapter 11 concludes this thesis. It contains a summary and possible extensions of this thesis.

1.5. Some Guidelines for the Reader

In this thesis the conceptual and implementational concepts are presented at object modelling level using UML [21]. Classes which appear as gray boxes are presented in another figure. See Figures 1.1 and 1.2 for a short introduction to UML notation as far as it is relevant for this thesis.

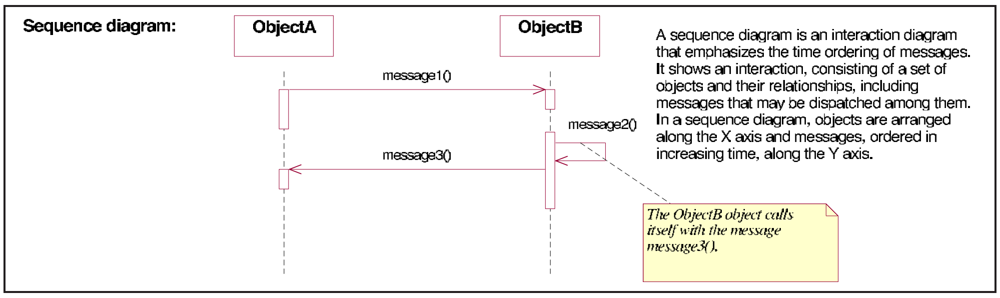
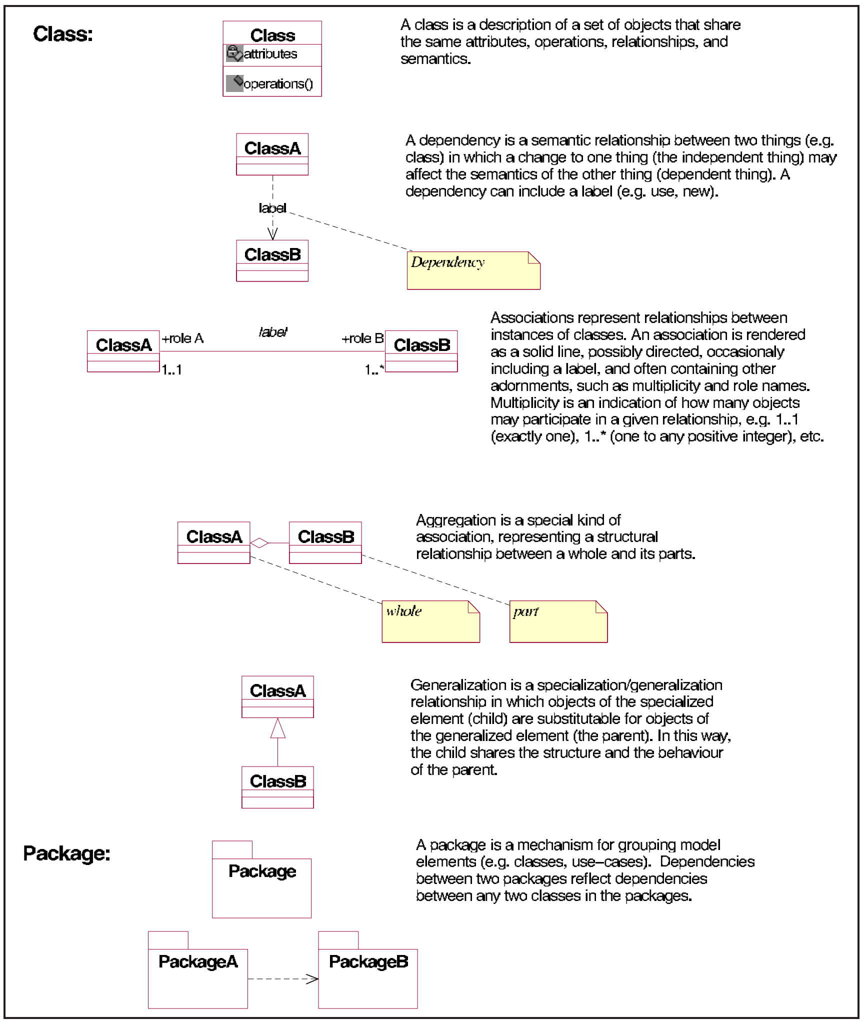

Sequence diagram:
ObjectA
ObjectB
message1()
message2()
message3()
A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. It shows an interaction, consisting of a set of objects and their relationships, including messages that may be dispatched among them. In a sequence diagram, objects are arranged along the X axis and messages, ordered in increasing time, along the Y axis.
The ObjectB object calls itself with the message message3().

Figure 1.1.: UML notation for sequence diagrams



A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics.

A dependency is a semantic relationship between two things (e.g. class) in which a change to one thing (the independent thing) may affect the semantics of the other thing (dependent thing). A dependency can include a label (e.g. use, new).

Associations represent relationships between instances of classes. An association is rendered as a solid line, possibly directed, occasionaly including a label, and often containing other adornments, such as multiplicity and role names. Multiplicity is an indication of how many objects may participate in a given relationship, e.g. 1..1 (exactly one), 1..* (one to any positive integer), etc.

Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

Generalization is a specialization/generalization relationship in which objects of the specialized element (child) are substitutable for objects of the generalized element (the parent). In this way, the child shares the structure and the behaviour of the parent.

A package is a mechanism for grouping model elements (e.g. classes, use–cases). Dependencies between two packages reflect dependencies between any two classes in the packages.

Figure 1.2.: UML notation for classes, packages, and relationships

2. Technical Background and Fundamentals

This chapter positions this thesis into the context of security technology and introduces the fundamentals – related to the security technology and distributed information systems technology – which are used throughout the thesis.

The rest of this chapter is structured as follows: Section 2.1 gives an introduction to authentication mechanisms. Section 2.2 introduces identity-based public-key infrastructures, in particular, the X.509 approach. Section 2.3 gives a short survey of basic access control approaches. Section 2.4 discusses the key-oriented trust management approaches PolicyMaker, KeyNote and SPKI/SDSI. In Section 2.5, we give a short introduction to the Knowledge Query and Manipulation Language, KQML. Finally, in Section 2.6, we briefly overview the ODMG object model and the object languages ODL and OQL.

2.1. Authentication

Authentication is the act of verifying a claimed identity. Authentication allows one entity as verifier to gain assurance that the identity of another entity as claimant is as declared, thereby preventing impersonation. Currently, most services use an authentication mechanism which is based on some secret only the claimant can generate. This secret may be a password that the claimant knows, a cryptographic key stored in a smart card of the claimant, or some biometric of the claimant (e.g. fingerprint, retina-scan, etc.). One can also combine two or more of the above approaches.

However, these approaches are geared towards human users. For the authentication between two processes running on two different hosts connected by an insecure communication channel, we need other authentication techniques supporting direct remote authentication. It might be the case that one process is running on behalf of a user. In this kind of authentication, the claimant has to prove knowledge of some secret (e.g. a cryptographic key, a password, etc.) that may be shared between the claimant and the verifier. The techniques which could be employed for this kind of authentication include *one-time passwords*, *challenge-response protocols*, etc.

According to the approach of one-time passwords [44], users can authenticate themselves to a remote system without the worry of that password being snooped and reused to gain unauthorized access to that system by an intruder. Once a one-time password is successfully used, the system will expect a new one-time password the next time, so although a one-time password can be snooped, it cannot be reused. The authentication systems designed for one-time passwords use generally a secret pass-phrase to generate a sequence of one-time passwords. With these systems, the user's secret pass-phrase never needs to cross the network at any time such as during authentication or pass-phrase changes. Thus, it is not vulnerable to replay attacks. Added security is provided by the property that no secret information needs to be stored on any system, including the server being protected.

In the case of using challenge-response protocols [72], a verifier has to *challenge* the claimant to prove that she holds the matching private key. Usually, the proof is accomplished by an appropriate *response* which is generated with the matching private key. A more sophisticated version of authentification is achieved when the claimant (key holder) performs a *zero-knowledge proof* of knowledge (about the private key) [37, 42] against the verifier.

Authentication schemes, such as Kerberos [65], that are based on *symmetric cryptography* requires two parties to share a secret key. In a distributed system, each pair of entities needs a different key. Obviously, this doesn't scale well. To master this challenge, the approach of *key distribution center* (KDC) has been introduced [64]. According to this approach, the KDC shares a different so-called *master-key* with every entity in the system. The KDC acts as a trusted intermediary in the authentication of two entities. During an authentication process, a session key is established to allow the two parties to communicate securely. In a distributed system that has K entities, using KDC reduces the number of required secret keys from $K(K-1)/2$ to K.

Secret-key based authentication schemes have some limitations. The KDC must always be on-line and highly available, because it is needed for every authentication. As a secret-key based system, Kerberos has also limitations, especially in the protocols and also in the whole design. Kerberos has no redundancy, i.e., if one server is compromised, the whole system is undermined. Kerberos requires synchronous communication with the servers, which can cause low performance. Other limitations related to Kerberos are presented in [12]. Secret-key based authentication schemes may serve well for a local-area network environment. However, it does not scale to the size of the Internet.

Compared to secret-key based authentication scheme, public-key cryptography systems make key distribution easier. In these systems, every entity is represented by a public/private key pair. Also known as asymmetric keys, the public/private pair is generated by independent but mathematically related algorithms. While an entity must keep the private part strictly hidden, the entity uses the matching public part as a visible surrogate. Thus, the public keys can be distributed freely. Public keys are good for proving the integrity and authenticity of messages from the represented entity, as well as for sending confidential messages to the represented entity. The verifier can easily authenticate a claimant, if the verifier knows the public key of the claimant. However, to be sure that a public key actually belongs to a particular entity (claimant), trusted certification authorities are still needed. A certification authority (CA) issue digital certificates. A certificate is a digitally signed message from a CA binding a public key to some property (e.g. name, personal attributes) pertaining to an entity which is the holder of the private key corresponding to the public key contained in the certificate. Certificates can be passed around, or managed in directories. Unlike KDCs, a CA can stay off-line.

2.2. Identity-based Public-Key Infrastructures

As stated above, a property assignment to an entity in the real world is presumably captured by a digital certificate issued by a trusted CA. To enable the authentication of entities in a distributed computing system, one CA is not enough. Asymmetric cryptography needs to be founded in an appropriate management of trust. In various forms, trust has to be assigned to public keys and to the pertinent properties that are claimed to be true for a public key or the holder of the corresponding private key, respectively. Management of trust is organized within a so-called *public-key infrastructure*, PKI for short. Elements of a PKI include a naming scheme of the property that is bound to public keys, data structures of certificates, mechanisms to distribute and revoke these certificates, and the intended semantics of certificates, which might be described in some certification policies. There are two important design questions for any PKI. The first question concerns the property assignment: "what information is bound to public keys?" A certificate states an ideal claim that do not necessarily hold. Thus, we are led to the second question which concerns the evaluation of trust about such ideal claims: "does a certificate capture a 'real' property assignment?" Earlier PKI proposals, such as X.509 [4, 51], Internet Privacy Enhanced Mail (PEM) [56] and Pretty Good Privacy (PGP) [98] are designed for authentication and bind a (hopefully) unique identity to each public key.

The ITU-T Recommendation X.509 (a de facto standard) defines a framework for the provision of authentication services, under a central control paradigm. X.509 focuses on defining a mechanism, by which information can be made available in a secure manner to a third-party. In the basic design of X.509, a certificate binds a public key to a distinguished name (DN) of the key holder. By issuing such a certificate, a CA assures third parties that some measure of care was taken into account to ensure that this binding is valid for both the name and key. However, the issue of whether a user's distinguished name actually corresponds to identity information which are linked to a person or simply to an e-mail address – and how such association was verified – is outside the scope of X.509 and depends on each certification authority's self-defined Certificate Practice Statement (CPS). A characteristic of X.509 is that it predicates that almost all issues which involve semantics or trust are delegated to a certification authority's CPS which is declared out of scope in relation to X.509. The certification authority's CPS is the governing law which the certification authority presents to potential clients and represents a top-down framework.

The X.509 approach employs multiple CAs which are arranged in a global hierarchy with one (or few) top-level roots (e.g. Verisign). In such a global hierarchy, CAs only certify their children and the top-level CA is the source of all certification paths. When two users certified by different CAs want to authenticate each other, a chain of certificates is used. When a chain of certificates is used to determine whether a certificate captures a "real" property assignment, the verifier has to trust all the CA's along the chain in which they are all honest and competent to make correct bindings and have followed good practice to generate and to sign the certificate. As the verifier may not know all the CA's, the trust can only come from the certificate chain. A certificate issued from a CA to another CA implicitly suggests the trustworthiness of the latter CA. Seen from the technical and policicital points of view, it is improbable to agree on a single trust point (the root CA), and is thus difficult to implement a global hierarchy.

Attempts have been made to extend the hierarchical CA model to interdomain authentication, but none has had much success beyond occasional biliteral cross-certification between two hierarchies, and thus between two CAs residing in different subtrees of the global hierarchy [88, 53]. Since hierarchical PKIs have unidirectional trust relationships, certification paths are easy to develop.

Privacy Enhanced Mail is a draft Internet standard for securing the Internet e-mail system. PEM employs X.509 certificates and implements a hierarchy of CAs.

PGP provides cryptographic routines for e-mail and file storage. In PGP, a user is identified by a *User ID*, which is an e-mail address followed by a name. A PGP certificate binds a public key to a User ID. A user can obtain and verify the public key of another user through *introducers* or some secure channels which may be outside cyberspace. An introducer asserts the binding between a User ID and a public key by signing this binding. Contrary to the PEM's strict hierarchy model, PGP follows a bottom-up approach where anyone can "certify" a key. In PGP, a "Web of trust" algorithm is used for determining when a binding is trusted, since every user can choose which introducers she trusts. In PGP, all keys and certificates are stored in key rings. Each user has a private key ring and a public-key ring. The former stores the public/private key pairs owned by the user and the latter stores public keys the user knows.

There are some problems that arise from the use of identity certificates [34, 22, 32]. One major problem is that the names of the certified individuals must be globally unique and unambigously recognizable by all those who rely on the certificates. It has proven difficult to create reliable global naming schemes.

2.3. Access Control

The purpose of an access control system is to enforce security policies by gating access to, and execution of, processes and services within a computing system via identification, authentication, and authorization processes. Thus, an access control system maintains confidentiality, integrity and availability by making it impractically hard for unauthorized entities to access resources.

Access control policies [77] can be grouped into three main categories: discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC).

DAC is a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. DAC permits the granting and revoking of access privileges to be left to the discretion of individual users who may *own* or create the objects. DAC enforces access control on the basis of the identity of the requesting subjects. A formal definition of discretionary access control usually includes the concept of an access control matrix. In an access control matrix, rows correspond to *subjects*, columns correspond to *objects*, and an entry indicates allowed *access rights*. Since storing a matrix as a two-dimensional array is a waste of memory space, some practical approaches have been developed to implement an access control matrix. Each column of the access matrix is stored with the object corresponding to that column, in the form of *access control lists* (ACL); each row of the access matrix is stored with the subject corresponding to that row, in the form of *capabilities*. Using ACLs requires the identification and authentication of the requesting subject. Granting capabilities requires the authentication of the recipient and precautionary actions against stolen and subsequently misused capabilities.

In the MAC approach, access control is based on *sensitivity* (as presented by a label) of the information contained in the objects and *clearances* of subjects which are willing to access information of such sensitivity. The most commonly used MAC is the multi-level security mechanism. The different security levels in a system form a lattice. In order to prevent a user from declassifying information without authorization, MAC is in general used to enforce one-directional information flow in such a lattice. A user with a given clearance level can only read information with the same or lower classification level. The rule for write access requires that a user with a given clearance level can only write information to a target object with the same or higher classification level.

Recently, RBAC [79, 36] has emerged as a promising alternative to the two traditional classes of access control policies stated above. The fundamental principal of RBAC is that the decision to allow access to objects is based on the role of the user. According to the core RBAC model, permissions are associated with roles, and users are granted membership in appropriate roles, thereby acquiring the roles' permissions. Access to a service requested by a user is granted, if the requesting user enjoys necessary roles associated with the requested service. The RBAC model also introduces the term *session*, which is a mapping between a user and an activated subset of roles that are assigned to the user. Roles are assigned to users based on the administratively-defined identities. More advanced RBAC models include also role hierarchies and constraints with respect to separation of duty relations. In a RBAC implementation, *user assignment* and *permission assignment* are two fundamental relations. Due to its representation of the access policy in two separate relations, the RBAC model lends itself naturally to the solution of access control problems in open distributed computing systems, such as *mediation*.

While some works [79, 36, 78, 77] use the terms *role* and *group* with different semantics, there are some proposals [81, 48] using these terms interchangeably. In [78], the authors discuss the distinctions among roles and groups. We agree with the authors of this work with respect to the following statements given for these distinctions.

Groups are named collections of users. By grouping the users in a group, we avoid the need of specifying a different authorization for each user. Authorizations assigned to a group hold for all the members of the group. As long as a user holds a group membership, she is allowed all the accesses that such membership implies. In contrast to a group, the role concept allows the specifications of authorizations that apply to users when performing some activities. Authorizations assigned to a role are applicable *only when* this role is *activated* by the user. Thus, using roles instead of groups allows the enforcement of the *least privilege principle* where each role is restricted to the execution of only those actions needed to perform a specific task.

A compact survey of basic access control approaches and other advanced proposals which refine, combine and compose these approaches can be found in [77].

2.4. Distributed Approach to Authorization and Trust Management

Traditional access control mechanisms operate under a closed world assumption, in which all of the entities are registered and locally known. New key-oriented[1] trust management systems generalize traditional mechanisms by eliminating the closed world assumption.

[1]In this thesis the terms *"key-oriented trust management approaches"* and *"capability-based trust management approaches"* are used interchangeably.

In a centralized system using traditional access control mechanisms, an identity often means an existing user account which is established with the system prior to the issue of any request. Earlier PKI proposals, such as X.509, try to establish a similar global user-account infrastructure as discussed in Section 2.2. In distributed computing systems, the very notion of *identity* becomes problematic. In a scenario in which an authorizer and a requestor have no prior relationship, knowing the requestor's identity may not help the authorizer make an access decision. Considering the problems that arise from the use of identity certificates [34] and due to the fact that centralized access control approaches do not match well the nature of authorization in open distributed systems like the Internet, one can argue that, in a global sytem, the only unique "identity" is the public key[2].

According to these trends, for general purpose authorization a number of key-oriented trust management systems have been developed. The key-oriented trust management systems PolicyMaker [17, 18] and KeyNote [16, 19] support the use of more expressive credentials that endow public keys with more than just identities. Public keys are treated as principals and authorized directly. Authorization is framed as a *proof-of-complicance* problem: "Does the set C of credentials prove that the *request r complies* with the local *policy P*?" These systems are designed to delegate asserted permissions, each of which acts as a capability giving the subject (i.e. which is denoted by a public key) certain *resource specific* permissions. In the PolicyMaker framework, there are policies and credentials, which together are referred to as *assertions*. An assertion contain predicates that describe the conditions under which one principal authorizes actions requested by other principals. Credential assertions and policy assertions have the same syntax. However, in contrast to policies, credentials are signed by their issuers as resource owners delegating the authority. The policy of a resource owner defines the root of all delegation chains. An application calls the PolicyMaker inference engine with a query and a set of policies and credentials. The inference engine provides an environment in which the policies and credentials can cooperate to produce a proof that the request complies with the policies in the case of a positive evaluation. Traditionally every application implemented its

[2]"*A public key (or its hash, if hashes are collision free) is a globally unique identifier of the private key. If proper key security is in place, as we must assume in all PKI systems, then the private key is associated with exactly one entity as keyholder. The public key or its hash is therefore a globally unique identifier of that entity.*" C. Ellison [30]

own mechanisms for specifiying access authorizations, binding user authentication to authorization and checking compliance. The developers of the trust management systems felt that this approach exposed the security of the application to high level of risk because the application developer might make simple but subtle mistakes when implementing the system. They propose to use an inference engine which is an off-the-shell security module that could be integrated into any application.

KeyNote was designed according to the same principles as PolicyMaker. In contrast to PolicyMaker, KeyNote requires assertions to be written in a specific language, while PolicyMaker assertions can be written in any programming language that can be safely interpreted by a local environment. In the KeyNote system, a calling application passes to a KeyNote evaluator a list of credentials, policies, requester public keys, and an "action environment", which consists of attribute/value pairs. The action environment is constructed by the calling application and comprises all information which is relevant to the request and necessary for the access decision. The included attributes and the assignment of their values must reflect the security requirements of the application accurately. The result of the evaluation is an application-defined string (e.g. "permitted") which is passed to the application.

The SPKI/SDSI approach [31, 35, 25] shares many views with the trust management approaches of PolicyMaker and KeyNote. SPKI and SDSI were started independently. Later, they merged into a collaborative effort SPKI/SDSI 2.0. Both approaches are motivated by the inadequacy of the public-key infrastructures based on global name hierarchies, such as X.509 and Privacy Enhanced Mail [56].

Contrary to KeyNote and PolicyMaker, SPKI/SDSI does not define an application-independent inference engine, since the authors in [33] state the processing of certificates and related objects to yield an authorization result as the province of the application developer.

SPKI/SDSI adopts SDSI's *localized naming scheme* [60, 25]. In SDSI, there are principals and local identifiers. Principals are public keys, and therefore unique. Each principal has its own local name space. Names are formed by linking principals and local identifiers. A local name is a principal followed by a local identifier, while an extended name is defined as a principal followed by at least two identifiers. A principal can use arbitrary local names and two principals might use the same name differently. SDSI allows the name spaces to be linked. Linking of name spaces allows principals to use definitions another principal has made. SPKI/SDSI has two kinds of certificates: *name certificates* and *authorization certificates*.

A name certificate binds a local name to a principal or an extended name. These certificates are used to resolve names to principals. Authorization certificates came originally from SPKI. An authorization certificate delegates a resource specific permission from a principal representing a resource owner as issuer or one of its delegatees to another principal (i.e. the entity being authorized in this certificate) stated in the certificate's subject field. The trust evaluation in SPKI consists of finding a delegation chain that delegates the authority from the original issuer to the principal encoded in the subject field of the final certificate of the chain. The capabilities being delegated are determined by the intersection of all the capabilities along the chain.

The common goal of the key-oriented approaches is to develop a standard form for digital certificates/credentials whose main purpose is authorization, rather than identity-based authentication. In other words, these new approaches obscure the boundary between security policy and certificates/credentials. The security policies that are traditionally maintained locally at the servers can now be encoded into the certificates/credentials and distributed to the remote entities. These approaches adopt a *peer model* of authorization, which means that every entity can be an authorizer, a third-party certificate issuer, or a requestor. An entity can act as a requestor in one authorization scenario and as an authorizer or a third-party certificate issuer in another.

In contrast to hierarchical PKIs, in the key-oriented approaches using a peer model of authorization, the path discovery becomes more difficult [25, 62]. Expressed more technically in terms of certificates/credentials, given a set of certificates/credentials and an access control list, it is non-trivial to determine whether a given principal (i.e. a public key) is authorized to access a protected resource.

The work in [90] presents a formal framework for expressing trust management systems with an application for KeyNote and SPKI.

In this thesis we followed and extended the key-centric view of authorization of the trust management systems, which states that the identities are neither necessary nor useful for authorization in open distributed information systems scenarios, since in such scenarios there is no relationship between a requestor and an authorizer prior to a request.

2.5. Knowledge Query and Manipulation Language

KQML [38], the Knowledge Query and Manipulation Language, is a language that is designed to support interaction among intelligent software agents. In a KQML-based agent architecture, the agents communicate by sending certain kinds of messages, called *performatives*, to each other. KQML performatives are the high-level “tell” and “ask” type operators on knowledge bases.

KQML is complementary to approaches to distributed computing, like CORBA [67], which focus on the transport level. In this thesis we employ CORBA for data exchanges and KQML for agent communications. KQML does not specify the content language used for cummunication. Therefore, two KQML speaking agents have to use any mutually agreeable content language. When using KQML, an agent transmits content messages, composed in the agreed content language, wrapped inside of a KQML language. In our agent-based architecture, we use OQL as a content language.

In the original version of the KQML [38], security issues were not taken into consideration. The works in [86] and [46] made some changes with respect to secure communications and PKI related communications. For the purposes of this thesis, we followed and extended the approaches presented in [86, 46].

2.6. ODMG Data Model and Object Languages

The common data model supported by ODMG [24] implementations is based on the OMG Object Model. The OMG core model was designed to be a common denominator for object request brokers, object database systems, object programming languages, and other applications. The ODMG data model provides a very rich set of modeling primitives. Objects are characterized by a state and a behavior. The state of an object is represented by the values of its properties. Properties can either be attributes or relationships. Relationships are declared between two object types and induce a pair of traversal paths between the two types. Moreover, the ODMG data model supports the notion of key, that is, a set of properties whose values uniquely identify each object within a class extent. A type is a specification which can have one or more implementations. A class is the combination of a type specification and a specific implementation. The model is strongly typed. The set of all instances of a type is the extent of the type. Abstract types are not instantiable.

They specify characteristics that can be inherited by subtypes but do not define any implementations. The model defines only type inheritance (i.e., subtyping). If S is a subtype of T, then S inherits all operations and properties of T, and S may define new operations and properties applicable to its instances. In other words, objects of type S have all the characteristics (and more) of type T. A subtype can specialize the properties and operations it inherits. A type can inherit from multiple supertypes, but must rename same-named inherited operations or properties.

The Object Definition Language (ODL) is a specification language used to define the object types that conform to the ODMG Object Model and is based on the OMG Interface Definition Language (IDL). ODL defines only the signatures of operations and does not address the definition of methods that implement these operations. ODL is intended to support the portability of database schemas. An ODMG schema is supportable by object-oriented database management system (e.g. O2) that conforms to the ODMG standard and any programming language for which there is an ODMG interface. The ODL constructions for defining schemas are incorporated in this interface. The Object Query Language (OQL) is the proposed query language of the ODMG object model. OQL is a declarative language that provides a rich environment for efficient querying of database objects. OQL is a strictly typed language permitting queries that deliver atomic or structured objects or literals as answers. Queries use collections for iteration purposes.

In this thesis we mostly follow the ODMG proposal to achieve interoperability among our agents with respect to schema declarations. In order to express query access authorizations related to schema declarations and corresponding instances, we extended ODL by adding two kinds of annotations to ODL schema declarations.

Part II.

Secure Mediation

3. Secure Mediation

3.1. Basic Mediation

Recent trends in information technologies have led to vastly improved communication facilities such as the Internet, an explosion of on-line information providers, and spontaneously emerging users. In the pervasive computing environments emerging from these trends, most interactions including business transactions occur between strangers, due to billions of spontaneous users and the fact that most of them do not share a common security domain. Complete strangers are willing to enter new business relationships and to conduct secure business transactions electronically, with a trust that comes close to traditional face-to-face business interactions.

More specifically, spontaneous users wish to find information and several heterogeneous and autonomous sources wish to share their resources and aim at supporting potential clients. Naturally, sources and clients will want to interact with each other in a confidential and privacy-preserving manner. Sources usually have to protect their information with respect to confidentiality, i.e., to take care to disclose information only to those clients who are entitled to see it. Clients may need to automatically verify that sources are certified to adhere to standardized, well-known privacy practices and procedures or that sources possess evidences proving that the sources are trustworthy and eligible.

Further on, while requesting accesses to the resources, clients may be unwilling to reveal their identities for private reasons and thus prefer to remain anonymous. Additionally for a resource owner, it may be necessary to see evidences of a client's eligibility rather than to know *who* they are. According to these trends, a client proves her eligibility to see a piece of information by a collection of her characterizing properties[1] (e.g. profession, organizational membership, security clearance, academic

[1] The personal identifying information pertaining to a client is also a characterizing property. In this chapter when using the term characterizing property, we mean the non-identifying properties.

title) and each autonomous source follows a security policy that is expressed in terms of characterizing properties.

When the resource owner and the requesting client are unknown to one another and when resources are to be shared across administrative boundaries, the conventional authorization scheme fails. The conventional approach to authorization involves authenticating the client to a local, administratively-defined user identity, then authorizing that user according to an access control list (ACL) for the resource. Given that the identities of clients cannot be known in advance to the receivers of the requests, these identities cannot be put on the access control lists and other identity-based authorizations of the receivers to decide who can do what.

In the simplest scenario we assume that the clients already know appropriate sources for the information they are looking for, and clients and sources do exchange messages directly. In more advanced scenarios, spontaneous users and on-line sources may have challenging demands. Whenever a user looks for a piece of information, she may aim at identifying promising sources which can be quite *heterogeneous* and *autonomous*. Whatever data a source has to offer, it may aim at supporting a *wide range* of potential clients, which are in general unknown in advance and may belong to *heterogeneous* and *autonomous* security domains. In order to meet these challenging demands, additionally *mediation* is used.

In mediated information systems [91, 95], a client seeking information and various autonomous sources holding potentially useful data, are brought together by a third kind of independent components, called *mediators*. Mediation is required to deal with heterogeneity and the autonomy of the sources, not only from the functional point of view but also with respect to all aspects of security, such as confidentiality and authenticity.

Considering the dynamics and the security requirements of mediation participants, we argue that the employment of merely identity-based authentication and authorization approaches is inadequate and ineffective for pervasive computing environments such as secure mediation. Seemingly, secure mediation requires a powerful and expressive *property-based* approach to authorization, taking the anonymity needs of the clients into account. This approach should be based on evidences of clients' eligibility rather than user authentication and access control based on user identities. We interpret the *identity* of an entity as a special kind of characterizing property identifying the entity. However, since the identities are not useful and not necessary for the authorization approach – as required by the secure mediation – the property-based approach to authorization should use a client's other characterizing properties rather than her identifying information.

Like in a marketplace of supply and demand, there exists different motivations for cooperation between the participants in mediation. The interoperability in mediated information systems may be stimulated by different participants having different motivations and security requirements. Depending on the motivation behind a stimulation, different kinds of mediators will be used for accomplishing different mediation processes, each of which has a different purpose for mediation. In the following we discuss two kinds of mediators and the corresponding mediation processes.

3.2. *I*-Mediation and *I*-Mediators

Clients demand systems enabling them to effectively work with heterogeneous information sources. This demand stimulates information sources to supply their information on an ad-hoc basis, in particular for purchase. Using mediators, clients can find and correlate information more efficiently. Information sources are likely to meet the client's requirements and to cooperate with mediators. Generally clients and information sources are independent of mediators. Clients are independent of information sources if there are several suppliers for their demand. Information sources exist in competitive as well as in non-competitive relationships with other information sources. Obviously there is no a priori base for direct mutual trust between the participants of a mediated system. In such scenarios, the mediation participants appear in the following roles:

- A *client* is characterized by her assigned properties and seeks for information.
- Each heterogeneous and autonomous *source* offers data and follows a security policy expressed in terms of characterizing properties.
- A *mediator* has two major functions:
    - From the functional point of view, the main role of a mediator is to retrieve, homogenize and assemble data from any sources the mediator may find worthwhile to contact.

- From the security point of view, the mediator contacts only the sources, whose security policies match a client's characterizing properties. Thus, seen from the client, a mediator acts as a kind of *filter* put in front of the sources.

We call such a mediator an *information integrating mediator*, also called an *i-mediator*. The process of mediation using *i*-mediators is called *i-mediation*. The owner of an *i*-mediator may want to maintain data owned by the *i*-mediator itself. Thus, an *i*-mediator can nearly be treated like a source. This makes *i*-mediation possible to be realized across several layers. As it is a source, an *i*-mediator way also want to protect its data with respect to confidentiality by following a security policy expressed in terms of characterizing properties.

3.3. *F*-Mediation and *F*-Mediators

As in the paper world, complex transactions in the electronic business world will involve asserted properties, commitments, etc. from many parties and the participants of such complex business transactions will, in general, not be in a position to understand or manage everything that is involved. They will need to trust domain experts and partner agents; they will also need to express delegation of authority over performing certain tasks to these trusted experts and agents.

More specifically, sources demand systems enabling them to determine a wide range of potential clients which could be interested in requesting specific services of the sources and which are qualified in terms of evidences of their eligibility. Information sources may supply their information for purchase as well as for collaboration. In the former case, the motivation of a source is to broaden its potential customer basis. In the latter case, a source's main motivation is to broaden its potential collaborators base by sharing its resources with the locally registered users of the potential remote partner-organizations, i.e., whose users are expected to be interested in the resources offered by the source.

The conceptual challenge arising in this situation concerns how remote and autonomous entities can agree on a common understanding of properties and other issues related to these properties (e.g. certificate formats) which are needed to decide whether an entity is eligible with respect to a protected service offered by another entity. On the one hand, properties are assigned to clients in their autonomously operating security domains, in principle without knowing their later usage. On the other hand, sources independently define their security policies in terms of these properties. In this situation the sources wish to be assisted to determine potential eligible clients.

To reach potentially eligible clients, a source could use a specific mediator having the required domain expertise as well as the relationships with the potential clients. A mediator could mediate between the security policy of a source and clients' characterizing properties which have been independently defined by the resource owner and asserted by some trusted parties vouching for the clients' characterizing properties, respectively. Sources could reach a wider range of potential clients by using such mediators.

In both cases, as stated at the beginning of this section, there is no a priori base for direct mutual trust between the participants of a mediated system. A main difference between the former and latter cases from a source's security policy viewpoint is that in the latter case, a source's policy might be based on the clients' characterizing properties which are related to the clients' organizational activities and responsibilities (e.g. organizational role membership or group membership) within the corresponding organizations to which the clients belong. The high level functionality common to both cases is *finding* potential clients and *stimulating* them to access resources.

By making some minor modifications to the scenario demonstrated above, it is possible to describe another scenario by shifting the role of a source as stimulator in the previous scenario to the client. A client may want to determine which (qualified) sources are willing to offer him a specific service and then request them. In this scenario clients demand systems enabling them to efficiently work with information sources which, on the one hand, may accept clients as potential eligible clients based on the client's asserted characterizing properties and, on the other hand, are the most appropriate sources according to the client's requirements with respect to the sources's expected qualifications, which may also be expressed in terms of characterizing properties, e.g., accredited hospital, authorized dealer.

Again, we are confronted with the same conceptual challenge described above. To master this conceptual challenge, a client could also employ a specific mediator having the required domain expertise as well as the relationships with the potential (qualified) sources. A mediator could mediate between the client's security requirements and characterizing properties, and sources' security policies and qualifications. Using such mediators clients can determine potential qualified and appropriate sources. In these scenarios there is again no a priori basis for direct mutual trust between the participants of a mediated system.

In the scenarios above, a mediator has the following major functions:

- From the functional point of view, the main role of a mediator is to seek out an entity B for another entity D and stimulate B to contact D.
- From the security point of view, the mediator acts as a broker between independently operating security domains of the mediation participants by mapping the properties and the security requirements of an entity B on the properties and the security requirements of the other entity D.

We call such a mediator an *entity finding mediator*, also called a *f-mediator*. The process of mediation using f-mediators is called *f-mediation*. The key concept of the f-mediation is the delegation of authorities. In f-mediation, an entity D *delegates* the authority over finding an eligible entity B to a f-mediator. Additionally, f-mediation can be transitively organized by using redelegation of received authorities. A f-mediator, as delegator, can express her trust in another f-mediator, as delegatee, to act as a delegator in turn.

The trustworthiness of a f-mediator may be determined based on the previous direct organizational or business relationships or on the recommendations or on the evidences of f-mediator's eligibility. For developing f-mediation, we have drawn our inspiration from the nature of real world relationships for which the delegation of authorities over performing certain tasks plays an important role.

3.4. Core Concepts

A new approach to authorization which is most suitable for secure mediation, according to the trends discussed above, necessitates a security framework which should have the following main characteristics:

Basic Characteristics:

I. An entity is represented by (one of) its public keys and characterized by the assigned properties.

II. An entity follows a security policy expressed in terms of characterizing properties.

Additional Characteristics for *i*-Mediation:

III. An entity E affirms that entity F has a certain property by assigning the properties of entity F to entity F's public keys.

IV. An entity E trusts, on its own discretion, entity F's judgements on a property.

Additional Characteristics for *f*-Mediation: In addition to the characteristics stated above, f-mediation still requires the following characteristics:

V. An entity E delegates the authority over evaluating a property to entity F, i.e., entity E trusts entity F's judgements on the property.

VI. An entity uses one property to make inferences about another property.

By combining an appropriate set of these characteristics, it is possible to design different approaches of secure mediation based on different trust models. Additionally, each particular design of secure mediation could be based on some other security requirements, with respect to all aspects of security. This includes confidentiality, authenticity, as well as integrity, anonymity, accountability and availability. These requirements must be fulfilled by appropriate security mechanisms, considering the dynamics and conflicting interests of the mediation participants of a particular design of secure mediation.

Considering the additional characteristics III and IV for i-mediation, the underlying basic informational environment of a design of i-mediation should be based on a public-key infrastructure supporting the trusted authorities who are assigning properties to the public keys of the entities. Considering the additional characteristics V and VI for f-mediation, the underlying basic informational environment of a design of f-mediation should be based on a trust management system supporting the delegation of authorities on performing certain tasks.

Though i-mediation and f-mediation are quite different in their detailed requirements and their purpose for mediation, they share a common requirement seen from the design and implementation points of view. Both mediation approaches need an *infrastructure for the management of trust* supporting property-based security policies.

The pure designs of widespread X.509 public-key infrastructure [4, 51] and the capability-based trust management systems, such as KeyNote [16] and SPKI/SDSI [31, 35, 25], are unfortunately inadequate for expressing property-based security policies as required by secure i-mediation and secure f-mediation. The explanatory statements why these systems are inadequate will be given in Section 10.2. In Chapter 4, we will present an infrastructure supporting property-based security policies.

As noted in Section 3.1, in general, properties need not to be identifying. Moreover, in distributed systems where entities want to cooperate without the knowledge of each other in advance, identifying properties are neither necessary nor useful (except possibly for tracing a detected fraud to a specific entity).

However, working within a distributed environment also requires authentification. Whenever an entity as issuer certifies a characterizing property for a subject, i.e., a public key, based on real world circumstances, the issuer must follow good practice in order to authenticate the public key. Also, when a resource owner decides on access to her services, based on a shown asserted characterizing property (i.e. which is encoded in a certificate or a credential), in most cases the owner must follow good practice in order to authenticate the subject, i.e., a public key. In both cases, the issuer or resource owner has to *challenge* the claiming entity to prove that she holds the matching private key. Usually, the proof is accomplished by an appropriate *response* which is generated with the matching private key. A more sophisticated version of authentification is achieved when the key holder performs a zero-knowledge proof of knowledge (about the private key) [37, 42] against the issuer or owner. Alternatively, if the issuer (but not the resource owner) is still thinking in terms of identities, she can also exploit an identification certificate with the considered public key as the subject together with the supporting licences.

3.5. Contributions

In this chapter we made the following contributions. We have classified the mediation as i-mediation and f-mediation according to different employment goals in distributed information systems. We have identified the main characteristics of a security framework which should establish a trust management basis for an expressive authorization approach which is to be suitable for secure i-mediation and f-mediation.

Part III.

Modelling a Hybrid PKI

4. A Hybrid PKI Model

4.1. Motivation

Though i-mediation and f-mediation presented in Sections 3.2 and 3.3, respectively, are quite different in their detailed requirements and their purpose for mediation, there may arise situations which allow the use of these approaches in a hybrid manner. In an instantiation of i-mediation, the corresponding i-mediator may serve spontaneous clients whose security domains are compatible with the i-mediator's security domain. Additionally, the same i-mediator might also serve clients which have already been stimulated by an f-mediator (i.e. which is acting on i-mediator's behalf) to request accesses from the i-mediator. In the latter case, the stimulation of a possible i-mediation task by any client requires an already completed f-mediation task, which has most probably been stimulated by the corresponding i-mediator. In this case, situations might arise where the i-mediator may want to serve some potential clients hosting in security domains which are incompatible with the i-mediator's security domain. To reach such clients, the i-mediator delegates its authority over the determination of users based on the users' characterizing properties to some f-mediators. A hybrid usage of i-mediation and f-mediation, as discussed above, is called *hybrid mediation*.

In this chapter we present a model of trusted authorities and licencing and a model of owners and delegation, which are designed to be fit for i-mediation and f-mediation, respectively. The employments of hybrid mediation and f-mediation requires both models to be integrated. For this purpose, we introduce a *hybrid PKI model*. The hybrid model for a PKI presented in this chapter constitutes a far more detailed elaboration of the main characteristics listed in Section 3.4.

The rest of this chapter is organized as follows. In Section 4.2, we present the features of a basic PKI model. We emphasize the subtle distinctions between the usually hidden (real) world of property assignment and its visible representation, classify properties of clients and servers and discuss the role of administrative properties. Section 4.3 summarizes how trusted authorities deal with so-called free properties (which include personal authorization attributes) using licencing if needed. Section 4.4 deals with the owners of servers and how they delegate part of their responsibilities, in particular for granting so-called bound properties (which include capabilities) for their services. The full hybrid model is presented in Section 4.5. Section 4.5.1 elaborates the central feature of the hybrid model which allows to convert authority based free attributes into owner managed bound properties for servers. Section 4.5.2 assembles the constituting parts into the full hybrid model. In Section 4.5.3, we present the classification of entities involved in an instance of the hybrid model. Section 4.6 evaluates the hybrid model against the core concepts of the secure mediation. In Section 4.7, we summarize the contributions made in this chapter.

4.2. Basic PKI Model

Asymmetric cryptography needs to be founded in an appropriate management of trust. In various forms, trust has to be assigned to public keys and to the pertinent properties that are claimed to be true for a public key or the holder of the corresponding private key, respectively. Management of trust is organized within a so-called *public-key infrastructure*, PKI for short. For any PKI, we have to consider requirements on quite different levels, ranging from legal rules, such as the European directive for digital signatures [71], over conceptual issues to final implementations. Here we concentrate on conceptual issues in regard to assigning properties and evaluating trust.

In its simplest form, a PKI is a system for publishing the public keys used in asymmetric cryptography. *Property assignment* and *trust evaluation* are two basic functionalities common to all PKI approaches. Property assignment is the process of binding a computer or other entity, or even some other piece of information, such as a capability, to a public key. Property assignment is captured by a *digitally signed document* which might be a *certificate* or a *credential*. Trust evaluation is the process of determining whether a digital document captures a “real” property assignment.

4.2.1. Property Assignment and Digital Documents

In a distributed system, a specific entity cannot directly see the other entities. In some sense, the (real) world of the other entities is hidden behind the interface to the communication lines. Surely, the entity can send and receive messages to and from this hidden (real) world. Based on these messages, the entity can produce a virtual view which is actually visible to the entity. As a consequence, security policies and permission decisions are solely grounded on the locally available visible view on the global (real) world. This outlined exposition is visualized in Figure 4.1.

An identifiable *entity* in the (real) world might be an *individual*, a *computer* or something similar that can act in the distributed system. An entity may *enjoy* various *properties* which might be relevant for security policies and permission decisions. In most cases, such properties are *assigned* to an entity by another entity.

In general, neither the entities themselves nor their properties are visible to other entities. Thus we need a notifiable representation of such circumstances. In a public-key infrastructure, PKI, entities are represented by one or more keys for asymmetric cryptography. More precisely, an entity is uniquely represented and distinguishable from other entities by one or more pairs of *private* and *public keys*: while the entity must keep the private part strictly hidden, the entity uses the matching public part as a visible surrogate for itself. From the perspective of the visible virtual views, these surrogates are called *principals*. In general, an entity may possess several key pairs. For the sake of conciseness, here we assume that each entity has exactly one key pair for digital signatures and one key pair for ciphers. Hence, here a principal is specified by a *public key for verification*, which is good for proving the integrity and authenticity of messages from the represented entity, and by a *public key for encryption*[1], which is good for sending confidential messages to the represented entity.

Then, a property assignment to an entity in the (real) world is *presumably_ captured_ by* a *digital document* in the visible virtual world. Such a document is called a *certificate* or a *credential*, depending on the details explained below. In all cases, the document has at least the following fields:

- a *subject* field which contains the principal, which visibly represents the entity under consideration;

[1]In practice, for efficiency reasons, usually a hybrid encryption scheme is exploited.

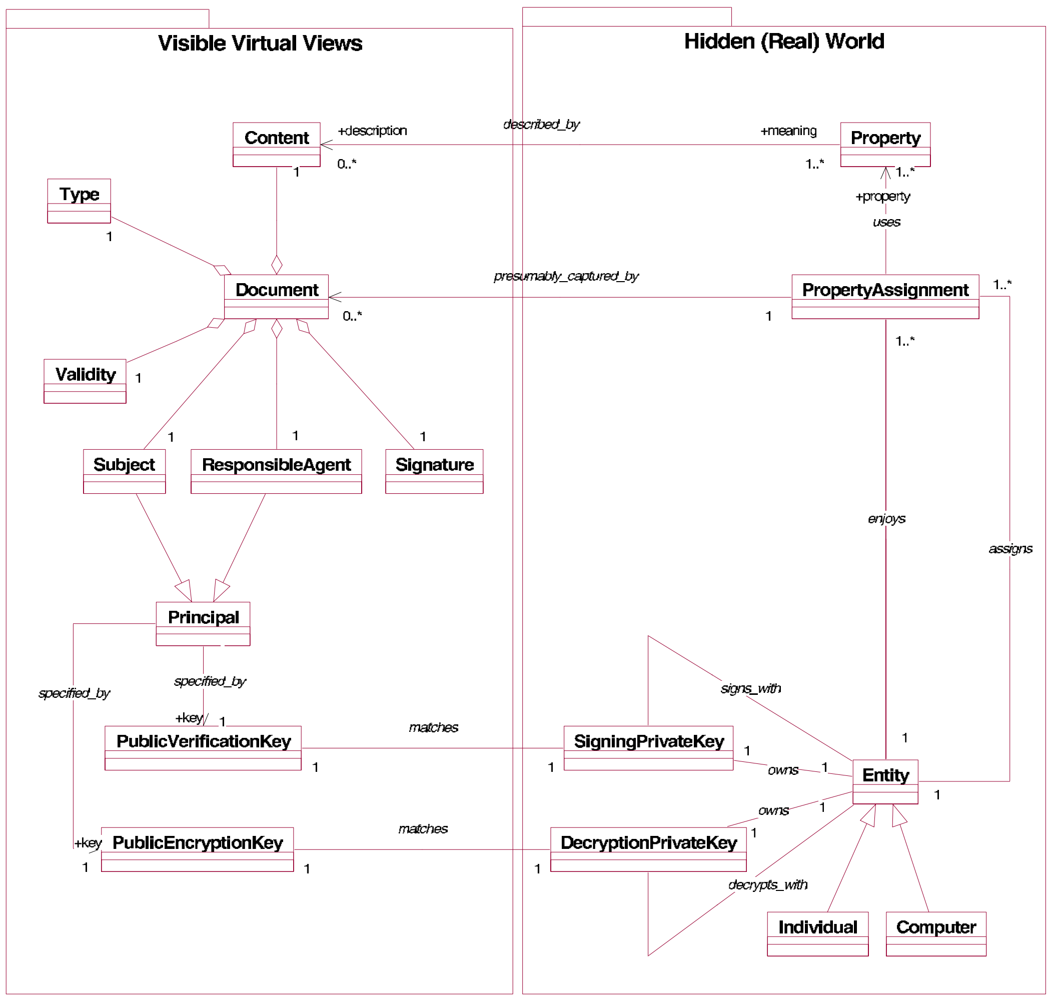


Figure 4.1.: Two faces of a property assignment process: hidden (real) world and visible virtual views

- a *content* field which textually describes the assigned property (where, depending on the the concrete format, we can also allow compound properties);
- a field for a *responsible agent*: this field contains the principal which visibly represents the entity that is responsible for the property assignment and that has generated and digitally signed the document;
- a *signature* field which contains a digital signature for the document: the signature is valid iff it can be verified with the responsible agent's public key for verification, i.e., if it has been generated with the matching private key for signing[2].

Depending on the specific format for digital documents, usually additional fields are needed, for example:

- a *type* field which indicates the meaning of the document and provides further technical hints on how to process the document;
- a *validity* field which might indicate that the responsible entity limits the property assignment to a certain time period or that the responsible entity otherwise restricts the usability of the document[3].

In the following sections, we classify properties of entities and thus contents of certificates and credentials according to two aspects. The first aspect deals with properties which characterize entities with security policies in mind. The second aspect touches upon the administration of characterizing properties.

4.2.2. Characterizing Properties

We distinguish two kinds of such characterizing properties, which are illustrated in Figure 4.2:

- A *free property* is intended to express some feature of an entity by itself (e.g. personal data, a technical detail, a skill, an ability). Other entities may possibly

[2] In this thesis, for the sake of simplicity, we assume that keys do not expire.

[3] Accordingly, again for the sake of simplicity, we also assume that digital documents are always usable and are never revocated.

base their security policies and permission decisions on shown free properties, but in general they have not expressed any obligation whether or how to do so. In particular, enjoying a free property usually does not entail a guarantee to get the permission for a specific service.

- A *bound property* is intended to express some relationship between a client entity and another entity which might act as a server (e.g. a ticket, a capability, a role). Typically, such a server has declared in advance that it will recognize a shown bound property as the permission to use some of its services. In particular, enjoying a bound property entails the promise to get a specific service as expressed in the relationship.

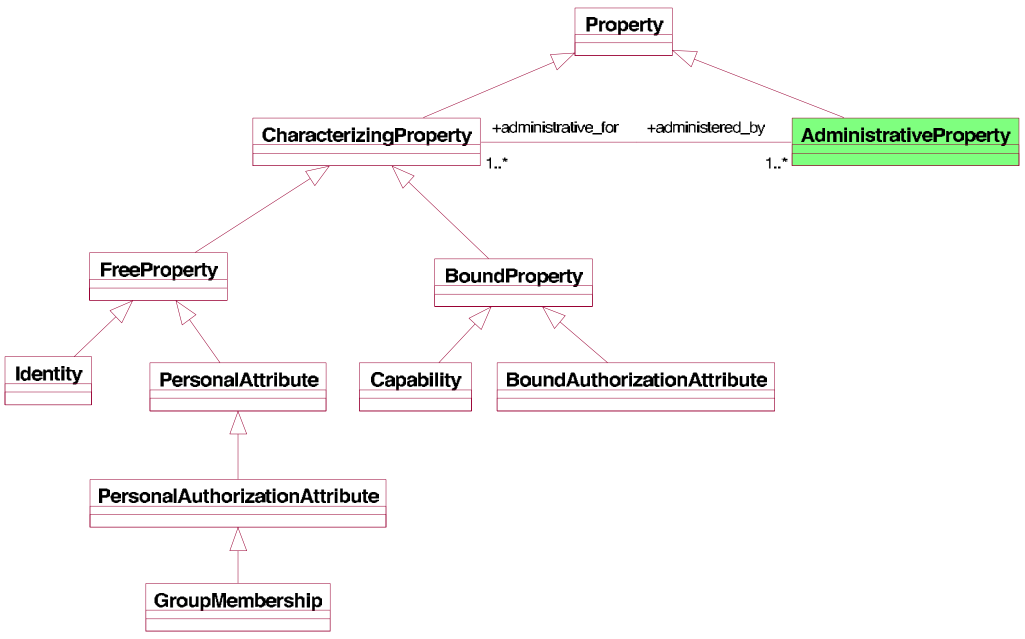


Figure 4.2.: A coarse classification of characterizing properties

Though in most situations the distinctions between the two kinds of characterizing properties are more or less obvious, they are difficult to be fully described generically. Rather than trying to do so, we are mainly interested in the analysis of how an entity can exploit its free properties in order to acquire bound properties, or in other words, how a client entity can convince a server entity to grant him the permission to use some services. This analysis is given in Section 4.5.

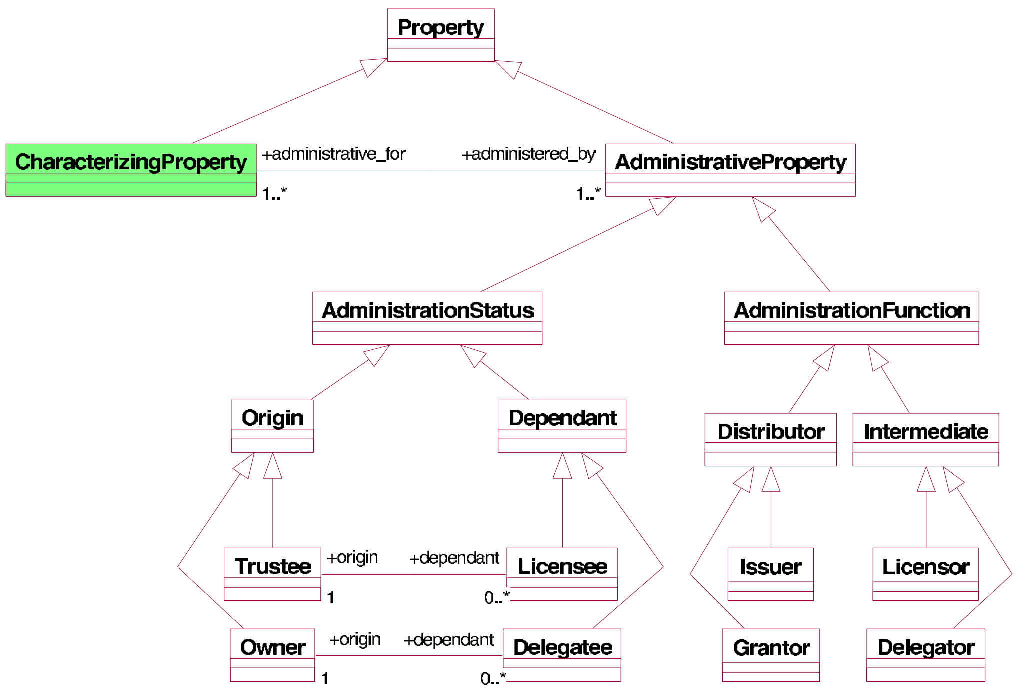


Figure 4.3.: A coarse classification of administrative properties

4.2.3. Administrative Properties

The assignment of characterizing properties to entities is regulated by corresponding administrative properties which must be hold by the entity that is responsible for such an assignment. We distinguish two kinds of administrative properties, which are depicted in Figure 4.3:

- The *administration status* expresses whether an entity can make assignments in its own right or only on behalf of another entity. In the former case, the assigning entity has the status of an *origin* (for the administered characterizing property), and in the latter case, it is considered as a *dependant* (for the administered characterizing property). The relationship between an origin and its direct or indirect dependants must be suitably expressed, again by appropriate administrative properties.

- The *adminstration function* expresses the following roles. In the role of a *distributor*, an entity can responsibly assign the corresponding characterizing property to a qualifying entity. In the role of an *intermediate*, an entity can establish new dependants.

Though administrative properties look quite similar for free properties and for bound properties, there are also some differences explained in Section 4.3 and Section 4.4 below. Moreover, usually different terms are used. For free properties, an origin is called a *trusted authority*, or *trustee* for short, and a dependant a *licensee*; for bound properties, an origin is seen as an *owner* (of a service) and a dependant as a *delegatee*. Accordingly, a distributor is an *issuer* or a *grantor*, respectively, and an intermediate is a *licensor* or a *delegator*, respectively.

We emphasize that all (potentially hidden) properties of entities, whether characterizing or administrative properties, have to be represented in the visible virtual world of certificates and credentials. The specialization hierarchy for properties is mirrored in corresponding specialization hierarchies for certificates and credentials which are exhibited in Section 4.3.2 and Section 4.4.2, respectively. As an important example, a (potentially hidden) relationship between an origin (which is administrative for a specific characterizing property) and its dependants is visibly reflected by an appropriate chain (or more generally an appropriate acyclic directed graph, *dag*) of certificates or credentials.

It is important to remark that the assignment of a property to an entity does not always signify the assignment of *trust* to the same entity. Solely, the assignment of an administrative property to an entity implies an assignment of *real world trust* to the same entity. In this case, the virtual representation of the real world trust may be called *digital trust* and is represented in a digital document. A characterizing property is assigned to an entity due to the fact that the entity enjoys a "real" characterizing property, not because the assigning entity implicitly trusts the entity enjoying the characterizing property.

4.2.4. Evaluating Trust

The relationships of *presumably_ captured_ by* are ideal claims that do not necessarily hold. A specific entity has to evaluate his individual trust about such an ideal claim. More specifically and among others, the specific entity seeing a document must evaluate his trust with respect to the following issues:

- Has the supposed assigning entity followed good practice to generate and to sign the document?
- Are the principals (keys) appearing in the document representing the supposed entities?

The very purpose of the administrative properties, and the corresponding certificates and credentials and the gathering of them into appropriate chains or dags is just to provide a reliable foundation for such trust evaluations, as explained in the following.

Conceptually, permission decisions are intended to be based on *characterizing properties* of entities appearing as clients. However, since property assignments occur in the hidden (real) world, actually the permission decisions must be based on available and visible digital documents, the contents of which *mean* the respective characterizing properties. Consider any such document as a so-called "*main document*" from the point of view of an entity entitled to take a permission decision. Then, the question arises whether the literal meaning of the content is indeed valid in the hidden (real) world, i.e., whether the digital document *captures* a "real" property assignment. This question is answered using further "*supporting documents*" the contents of which mean appropriate administrative properties. However, for each of these supporting documents the same question arises: is the literal meaning of the content indeed valid in the hidden (real) world? Thus, we are running into a recursion:

- The "main document" concerning a characterizing property needed for a permission decision is supported by a first level of "supporting documents" concerning administrative properties for the characterizing property.
- For each "supporting document" of the i-th level, one of the following cases holds:
  - Either, it is supported by further "supporting documents" of the next level, thereby expressing that the responsible agent of the former document (that must be identical with the subject of the latter documents) represents a *dependant* of the responsible agents of the latter documents.

- Or, it expresses that its responsible agent represents an *origin* for the administered characterizing property expressed by the content of the “main document”.

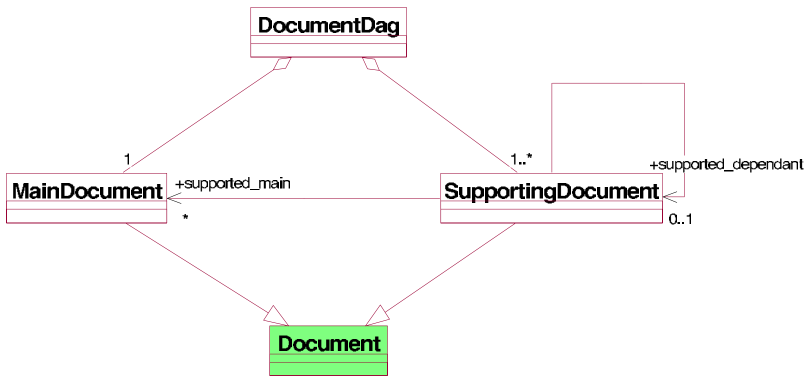


Figure 4.4.: Structure of document dags (and chains) which formalize the borderline relationships of *presumably_ captured_ by*

In order to be helpful, the “main document” and its “supporting documents” should form a *directed acyclic graph (dag)* with respect to their relationships concerning *support*. A corresponding class model is visualized by Figure 4.4, which can be understood as an extension of the left part (about the visible virtual views) of Figure 4.1. This extension is devoted to formalize the borderline relationship *presumably_ captured_ by* strictly within the visible virtual views. As a special case, we just obtain a *chain*. For example, Figure 4.9 (see Section 4.5) shows two chains, one for a free property-certificate as the “main document”, and another one for a bound property-credential as the “main document”.

In any case, the ultimate trust about all ideal claims pertinent to the documents relies on the *nonsupported* documents referring to *origins*. Rather than using explicit documents for origins, the evaluating entity often just decides on its own discretion that it wants to treat the responsible agent of a “supporting document” as denoting an origin. This situation is shown in Figure 4.9, where no explicit documents occur for origins, i.e., the trustee and the owner, respectively. A main difference between administrative properties for free properties and for bound properties stems from different treatments of and assumptions on origins and how origins determine their dependants.

4.3. Model of Trusted Authorities and Licencing

Based on the fundamentals given in Section 4.2, we present in the following how property assignment and trust evaluation are performed in the model of trusted authorities and licencing. We also show how the certificates are classified according to their usage and contents.

4.3.1. Property Assignment and Trust Evaluation

Following and extending the basic approach of X.509 [4, 51], free properties and the corresponding certificates are handled by trusted authorities using licencing. Figure 4.5 visualizes an instance of the general situation. In the simplest case, an entity acts and may be considered as a "trusted authority" or trustee for a free property.

In *acting*, such an entity assigns a free property to another entity and accordingly *issues* a suitable certificate (called *free property-certificate*) about this assignment. In the certificate, the content expresses the free property, the subject is the principal (public key(s)) representing the enjoying entity (holder), and the responsible agent is the principal (public key(s)) representing the trustee (issuer). The integrity and authenticity of the certificate is assured by a digital signature which the issuer generates with his private key for signing.

In *being considered* as a "trusted authority", such an entity is afterwards evaluated by a further entity (verifier). This further entity, seeing the issued certificate, may decide to treat the issuing entity as an origin for the free property. Thus, after having verified the signature of the certificate, the further entity concludes his trust evaluation whether or not the certificate captures the corresponding property assignment. The crucial point here is that in general the issuer and the holder of the certificate are different from the entity which afterwards inspects the certificate.

In more advanced cases, additionally *licencing* is used. Then, basically, an entity engaged in licencing does not assign free properties on its own right. Rather, it has to be *explicitly licenced* to do so. More specifically, some other entity, acting as a *licensor* and trusting the *licensee*, has expressed by a *licence-certificate* that the licensee should be entitled to assign a specific free property, i.e., to issue corresponding certificates. In the licence-certificate, the content means trustworthiness to assign the specific free property, the subject is the principal (public key(s)) representing the

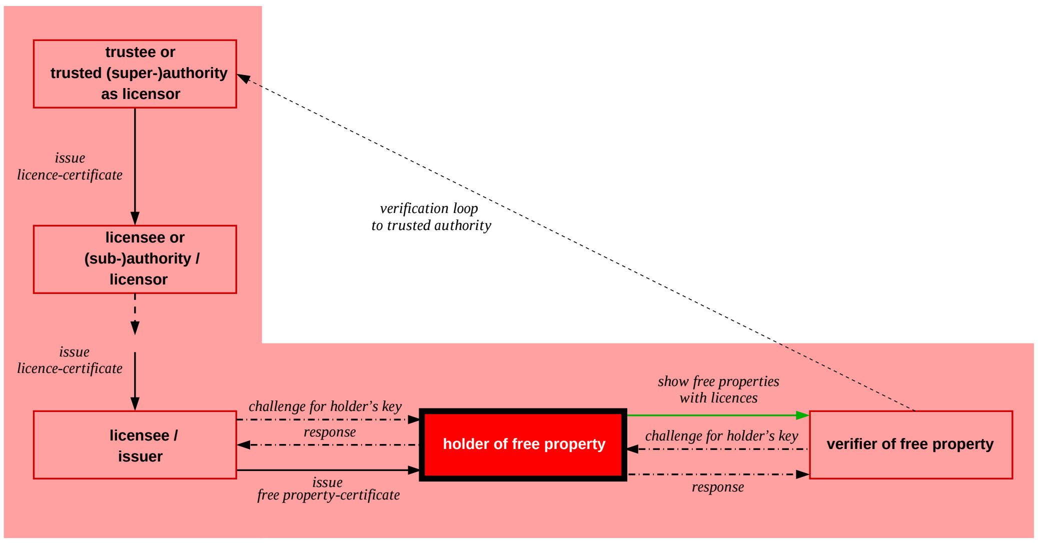


Figure 4.5.: Outline of an instance of the model of trusted authorities and licencing

licensee, while the responsible agent is the principal (public key(s)) representing the licensor.

Additionally, licencing can be transitively organized: a licensor can express his trust in a licensee to act as a licensor in turn, again by a suitable licence-certificate. Here, suitability depends on the application context: generally speaking, the precise scope of the licence must somehow be described. For instance, the licence to act as a licensor can be restricted to a fixed predetermined free property, or it can include to define new free properties of a certain kind. It might also be the case that a licensee needs to be trusted by several licensors.

Whenever an issuer certifies a free property for a subject, i.e., a public key, based on real world circumstances, the issuer must follow good practice in order to authenticate the public key. Also, when the verifier of free property wants to gain assurance whether the holder of a shown asserted free property is also the holder of the matching private key, in most cases the verifier must follow good practice in order to authenticate the subject, i.e., a public key. In both cases, the issuer or verifier has to *challenge* the claiming entity to prove that she holds the matching private key. Usually, the proof is accomplished by an appropriate *response* which is generated with the matching private key.

In this model of trusted authorities and licencing, a special kind of certificates occur, namely the *identity-certificates*[4] (see Section 4.3.2) which deal with free properties that can be used to identify an entity. Typically, such a free property is a name or some number (hopefully) uniquely used within a domain.

Identity-certificates are needed whenever, seeing a public key, there is an interest to identify the entity that possesses the key. We emphasize, however, that in many cases the identity of an entity is not important at all. Rather, the interest is directed to other free properties and who has been the issuers of these free properties. Again, the direct issuer's identity might also not be important; rather one might prefer to look on the issuer's licensors instead. Surely, in most cases, following the dag of supporting licence-certificates, the origins should be identifiable. Thus, normally they suitably publish identity-certificates which enable an evaluating entity to take his discretionary trust decisions.

A free property-certificate and its licence-certificates form a directed acyclic graph (dag). As a special case, we just obtain a *certificate chain*. Chain evaluation[5] is done in a similar way as it is described in Section 4.2.4.

4.3.2. Classification of Certificates

In Sections 4.2.2 and 4.2.3, we have presented two specialization hierarchies, one for free properties and another one for the corresponding administrative properties. In the following, we present a specialization hierarchy for corresponding certificates.

We classify certificates of entities according to whether they embody a characterizing property or an administrative property (see Figure 4.6). We distinguish two kinds of certificates.

A *characterizing certificate* might be a *free property-certificate* whose content means a *free property* (see Section 4.2.2). A free property-certificate might be an *identity-certificate* [3, 4], an *attribute-certificate* [4, 1, 73], an *accreditation-certificate*[6] [7, 39] or a *private-certificate*[7] [22].

An *identity-certificate* binds an identifying name – which should be (hopefully) unique within a domain – associated with an individual or a computer to a public

[4]The X.509 term is *public-key certificate.*

[5]X.509 uses the term *certification path validation.*

[6]In [7], we used the term *credential.* In this thesis we would prefer to call the document an *accreditation-certificate.*

[7]The work in [22] uses the term *private credentials.*

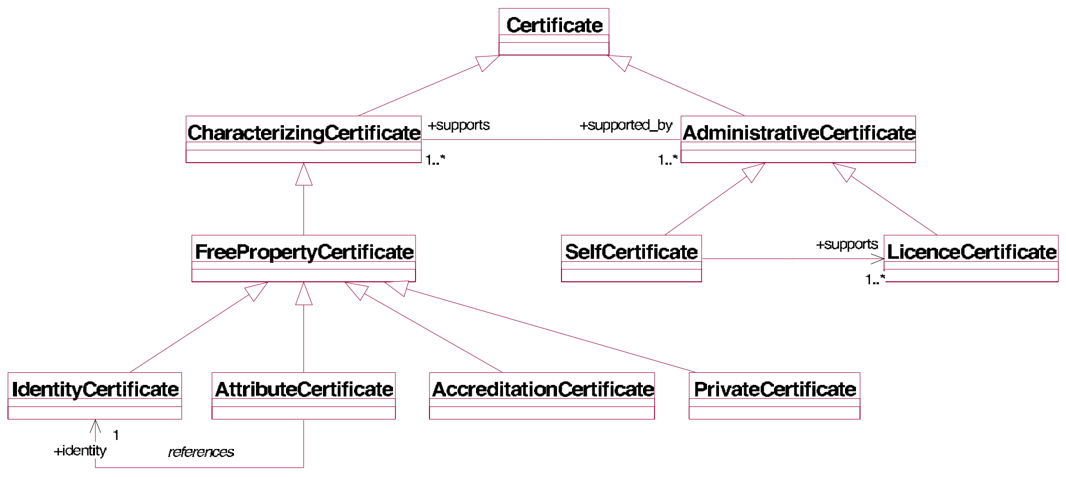


Figure 4.6.: Specializing hierarchy of certificates in the model of trusted authorities and licencing

key. In an identity-certificate, in which the content expresses the identified entity, the subject is the public key representing the identified entity, while the responsible agent is the public key representing the entity which certifies the key occuring as subject in the same identity-certificate. Identity certificates are not useful in many situations. In contrast to traditional proof of identity, identity certificates enable all of an entity's actions to be linked and traced automatically by other entities. The large use of identity certificates greatly increases the risk of identity fraud. See [22] for a detailed discussion of privacy dangers related to identity-certificates.

An *attribute-certificate* is conceptually similar to an identity-certificate in terms of format and philosophy. Instead of binding an identifying name to a public key, an attribute-certificate binds a personal attribute to a public key. The major difference with an identity-certificate is that attribute-certificates do not contain a public key, but they refer to an identity-certificate. For an evaluating entity to use this information, it must combine an attribute-certificate with an identity-certificate. In an attribute-certificate, the content means a personal attribute of an identified entity, the subject[8] is a reference to an identity-certificate holder which enjoys this personal attribute, and the responsible agent is the principal representing the en-

[8] In the class model visualized by Figure 4.1 (see Section 4.2.1), a principal is specified by a public key for verification and a public key for encryption. In order to represent an attribute certificate, the class *principal* must be further refined to handle the references to identity-certificates.

tity that verifies the referenced identity-certificate and, in the positive case, certifies the binding of the personal attribute to the key occuring as subject in the verified identity-certificate. Most likely, an identity-certificate and an attribute certificate are certified by different issuers, one an authority on the personal attribute and the other an authority on identifying names.

An *accreditation-certificate* binds also a personal attribute to a public key. In contrast to attribute-certificates, an accreditation-certificate is not associated with any identity-certificate. In an accreditation-certificate, the content expresses the personal attribute, the subject is the principal, i.e., the public key representing the personal attribute's holder (a hidden entity probably identified[9] by the trusted authority as issuer of the corresponding accreditation-certificate), and the responsible agent is the principal representing the entity that certifies the key occuring as subject. In some cases, the trusted authority as the issuer of an accreditation-certificate does know the mapping between the identified entity (probably via entity's identity-certificate) and accreditation-certificate. In such cases, the mapping needs not to be revealed to the evaluating entity when requesting a service from that entity. Only when investigating misuse of a service or behavior of an individual, the mapping should be revealed. The identity of an entity holding an accreditation-certificate remains hidden so far as the evaluating entity cannot infer the identity from its knowledge about the personal attributes, from the communication protocol data and the entity's request, as well as the answer corresponding to his request. In other cases, the trusted authority as the issuer of an accreditation-certificate may not know the mapping between the identified entity and accreditation-certificate. In such cases, the issuer issues an accreditation-certificate based on the other accreditation-certificates and the corresponding licence-certificates issued by other trusted authorities.

Private-certificates are proposed by Stefan Brands with a different conception and implementation of digital certificates, mainly that privacy is protected without sacrificing security. The validity of such certificates and their contents can be checked, but the identity of the certificate-holder cannot be extracted, and different actions by the same person cannot be linked. Certificate holders have control over what information is disclosed to the receipent. Although the certificate may contain a set of attributes, the holder may chose a subset of them to be accessible by the verifier

[9]The authentication of the public key is accomplished by using a challenge-response protocol (see Figure 4.5).

of the certificate. Stefan Brands' certificates are expressly anonymous.

An *administrative certificate* embodies an administrative property (see Section 4.2.3). An administrative certificate might be a *trustee self-certificate*[10] or a *licence-certificate*[11]. A trusted authority as trustee signs its own certificate, i.e., a trustee is self-signed. The resulting certificate is called *trustee self-certificate*. A trusted authority as licensee is certified by another trusted authority as licensor. In this case, the resulting certificate is called *licence-certificate*.

4.4. Model of Owners and Delegation

In the model of trusted authorities and licencing described in Section 4.3, it was solely possible for some trusted authorities to assign properties and to issue corresponding certificates to entities enjoying the assigned properties. Naturally, it should be possible for any entity, as owner of its resources, to define his own vocabulary for properties, to grant corresponding digital documents, and even, to express his trust in delegatees, each of which is entitled to assign a specific property (i.e. defined in the owner's vocabulary for properties) to other entites. These requirements are fulfilled by the model of owners and delegation.

Again, based on the fundamentals given in Section 4.2, we present in the following how property assignment and trust evaluation are done in the model of owners and delegation. We also show how the credentials are classified according to their usage and contents.

4.4.1. Property Assignment and Trust Evaluation

Following and extending the basic approach of SPKI/SDSI [31, 35, 25], bound properties[12] and the corresponding credentials[13] are handled by owners of services using delegation. Figure 4.7 visualizes an instance of the general situation. In the simplest case, an entity acts as an owner of its services (resources) offered to other entities and is explicitly addressed by these other entities.

[10] The X.509 term is *root certificate*.
[11] The X.509 term is *certification authority certificate*.
[12] SPKI/SDSI uses the term *authorization* instead of *bound property*.
[13] SPKI/SDSI uses synonymously the terms *authorization certificate* and *capability certificate*.

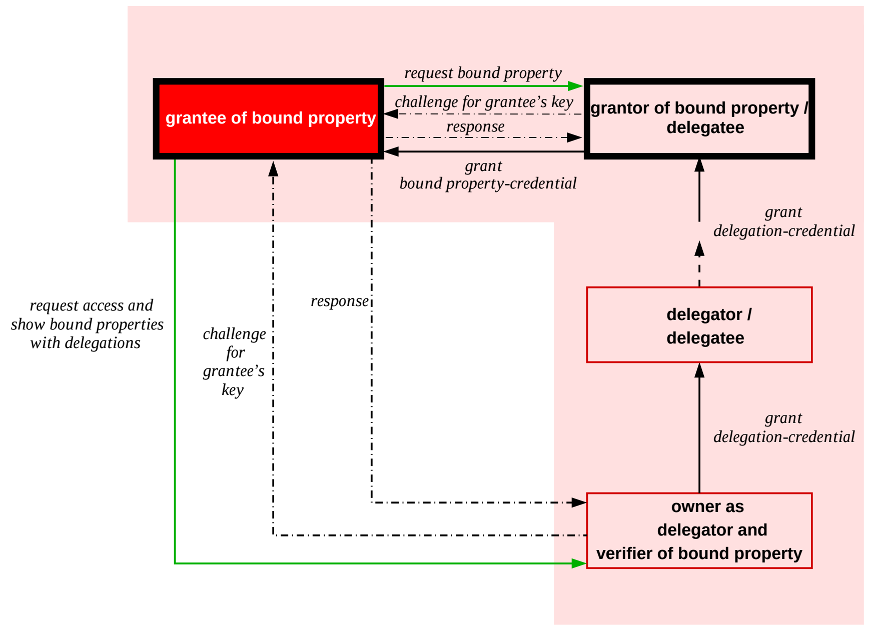
In *acting*, such an entity assigns a bound property to another entity and, accordingly, *grants* a suitable credential (*bound property-credential*) about this assignment. Like for a certificate, in the credential, the content means the bound property, the subject is the principal (public key(s)) representing the enjoying entity (grantee), and the responsible agent is the principal (public key(s)) representing the owner (grantor). Again, the integrity and authenticity of the credential is assured by a digital signature which the grantor generates with his private key for signing.

In *being addressed* as an owner, such an entity is afterwards contacted by some further entity which requests the offered service. The request comes along with showing a credential in order to get the permission to access the service. The addressed owner inspects the credential whether he himself has granted it, i.e., he checks the signature with his public key for verification. In the positive case, the owner interpretes the bound property expressed by the content of the credential according to his security policy:

- If the bound property is interpreted as a traditional *capability*, then the owner immediately allows access to the requested service provided the capability is good for the service. Thus, for a capability the essential permission decision has been taken previously at the time of granting the credential.
- If the bound property is interpreted as a more general *bound authorization attribute*, then the owner might base his final permission decision not only on the shown bound authorization attribute granted some time before, but also on additional factors which are not directly encoded in the credential.

It is important to note that, so far and neglecting misuse of stolen credentials, the owner needs no trust evaluation except that he is willing to accept his own signatures. The crucial point here is that the grantor of the credential is identical with the entity, which afterwards inspects the credential.

In more advanced cases, additionally *delegation* is used. In this case, an entity engaged in delegation does not assign bound properties on its own right and for its own services. Rather, it is acting on behalf and in explicit delegation of a different owner. More specifically, some other entity, acting as a *delegator* and trusting the *delegatee*, has expressed by a delegation-credential that the delegatee should be entitled to assign a specific bound property, i.e., to grant corresponding credentials.



request bound property
challenge for grantee's key
response
grantee of bound property
grantor of bound property / delegatee
grant bound property-credential
grant delegation-credential
request access and show bound properties with delegations
challenge for grantee's key
response
delegator / delegatee
grant delegation-credential
owner as delegator and verifier of bound property

Figure 4.7.: Outline of an instance of the model of owners and delegation

In the delegation-credential, the content means trustworthiness to assign the specific bound property, the subject is the principal (public key(s)) representing the delegatee, and the responsible agent is the principal (public key(s)) representing the delegator.

Additionally, delegation can be transitively organized: a delegator can express his trust in a delegatee to act as a delegator in turn, again by a suitable delegation-credential. It might also be that a delegatee needs to be trusted by several delegators.

Whenever the grantor of bound property grants a bound property for a subject, i.e., a public key, the grantor must follow good practice in order to authenticate the public key. Also, when a resource owner decides on a permission for his service based on a shown credential, in most cases the owner must follow good practice in order to authenticate the subject, i.e., a public key. In both cases, the grantor or resource owner has to *challenge* the claiming entity to prove that she holds the matching private key. Usually, the proof is accomplished by an appropriate *response* which is generated with the matching private key.

In this model of owners and delegation, if any bound property-credential is used as a “main document” then it is shown to the owner(s) of the services to which the bound property refers. In the case that a delegatee is trusted by only one delegator, any dag of supporting delegation-credentials contains just one origin, namely the owner himself. In some special cases, where a delegatee is trusted by several delegators, any dag of supporting delegation-credentials contains more than one origin, namely collaborating[14] owners themselves.

A bound property-credential and its delegation-credentials form a directed acyclic graph (dag). As a special case, we just obtain a *credential chain*. Chain evaluation[15] is done in a similar way as it is described in Section 4.2.4.

4.4.2. Classification of Credentials

In Sections 4.2.2 and 4.2.3, we have presented a specialization hierarchy for bound properties and the corresponding administrative properties, respectively. In the following we present a specialization hierarchy for corresponding credentials.

We classify credentials of entities according to whether they embody a characterizing property or an administrative property (see Figure 4.8). We distinguish three kinds of credentials.

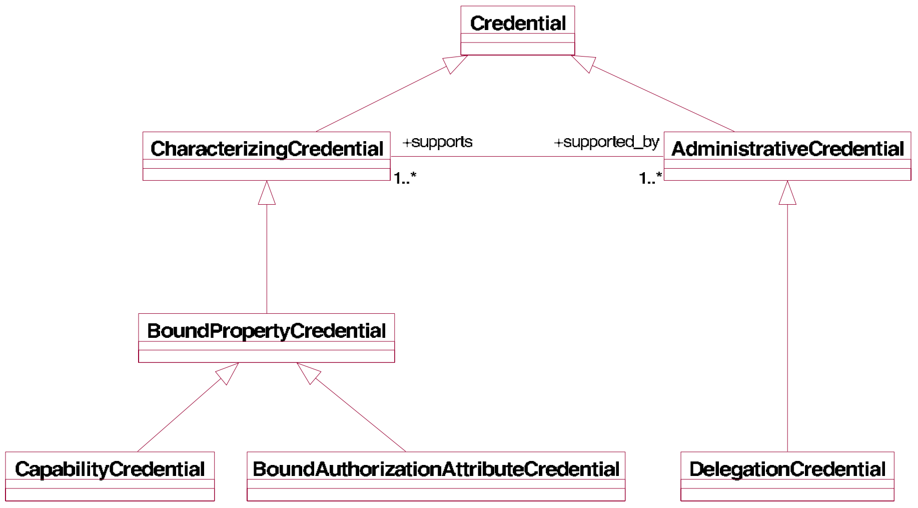


Figure 4.8.: Specializing hierarchy of credentials in the model of owners and delegation

A *characterizing credential* might be a *bound property-credential* whose content

[14] It might be the case that access to a specific resource is regulated by more than one owners.
[15] SPKI/SDSI uses the so-called *certificate reduction* mechanism to evaluate chains.

means a *bound property* (see Section 4.2.2). A bound property-credential might be a *capability-credential* or a *bound authorization attribute-credential*.

In a *capability-credential*, the content expresses a capability, the subject is the principal (public key(s)) representing the enjoying entity (grantee), and the responsible agent is the principal (public key(s)) representing the owner (grantor).

In a *bound authorization attribute-credential*, the content means a bound authorization attribute, the subject is the principal (public key(s)) representing the enjoying entity (grantee), and the responsible agent is the principal (public key(s)) representing the owner (grantor).

An *administrative credential* might be a *delegation-credential*. In a delegation-credential, in which the content means trustworthiness to assign the specific bound property, the subject is the principal (public key(s)) representing the delegatee, while the responsible agent is the principal (public key(s)) representing the delegator.

In the specific SPKI/SDSI approach, the assignments of characterizing properties and administrative properties are captured in the same credential. The value of a so-called delegation-bit determines whether or not a credential is interpreted as a delegation-credential. In this approach a delegatee can use the characterizing properties (i.e. which are encoded in the credential granted to him) to access the services associated with these properties for himself. In the cases where a delegatee is only authorized to grant characterizing properties to other eligible entities, it might be necessary to bind a delegatee to certain obligations with respect to granting bound property-credentials. In forthcoming disputes, a delegator can use these obligations to blame a delegatee for detected misuse. For such cases, we argue that distinguishing between administrative credentials and characterizing credentials is useful, since such obligations could be encoded in corresponding delegation-credentials.

4.5. The Full Hybrid Model

In the previous sections, we identified the similarities and the differences of two PKI approaches. Roughly summarized, they are similar with respect to the need of supporting documents for a main document, a certificate or a credential, respectively, but they differ with respect to how trust evaluations are performed. We argue that many applications require to use and to link both kinds of PKI approaches. In this section we outline a hybrid PKI model which unifies and extends the previous approaches.

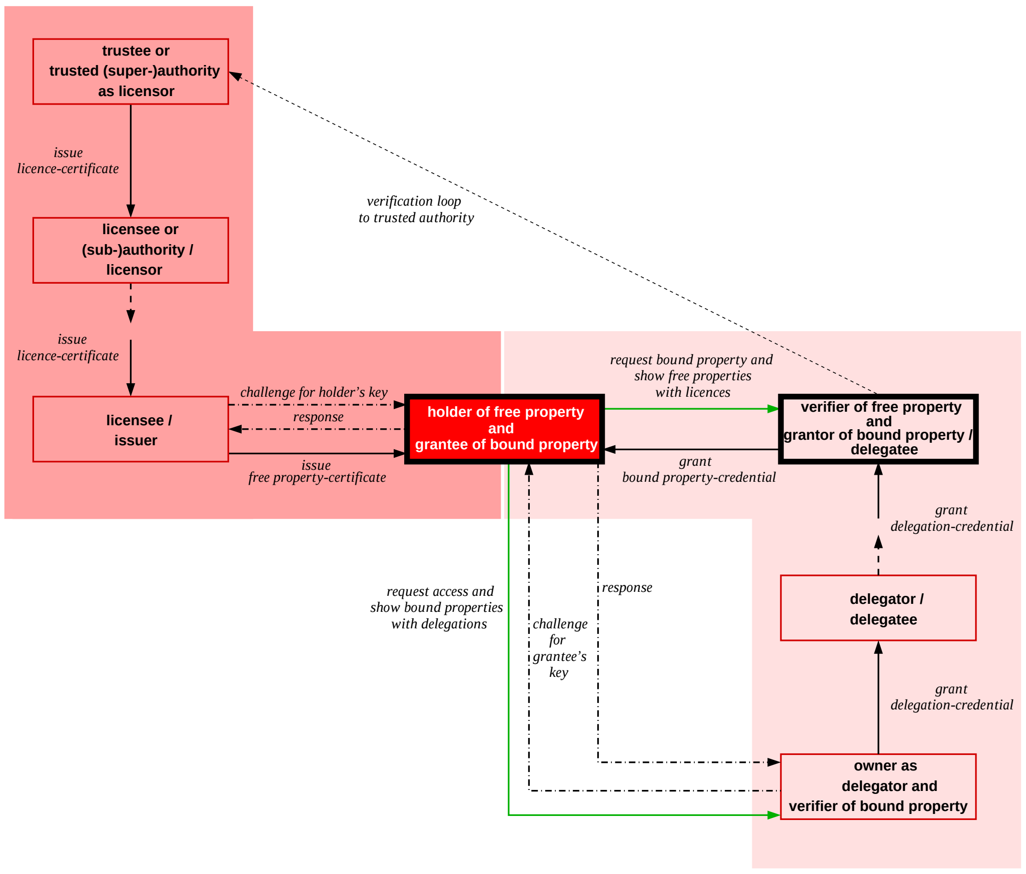

trustee or trusted (super-)authority as licensor
issue licence-certificate
licensee or (sub-)authority / licensor
issue licence-certificate
licensee / issuer
challenge for holder's key
response
issue free property-certificate
holder of free property and grantee of bound property
verification loop to trusted authority
request bound property and show free properties with licences
verifier of free property and grantor of bound property / delegatee
grant bound property-credential
grant delegation-credential
delegator / delegatee
grant delegation-credential
request access and show bound properties with delegations
challenge for grantee's key
response
owner as delegator and verifier of bound property

Figure 4.9.: Outline of an instance of the hybrid model for a PKI

4.5.1. Converting Free Properties into Bound Properties

Why is a grantor, whether the owner of a service himself or any of his delegatees, willing to assign a bound property to an entity and to grant a corresponding credential, i.e., to express a possibly conditional permission to access a service? The general answer is that the grantor follows a *property conversion policy* that maps free properties on bound properties, where the property conversion policy is a part of grantor's whole security policy. More precisely, the property conversion policy specifies

> *which set of free properties an entity has to enjoy in order to obtain a bound property assignment.*

Rephrased more technically in terms of the visible world of digital documents, this means the following: the property conversion policy specifies

> *which free property-certificates as "main document" together with which "supporting licence-certificates" are accepted in order to obtain which bound property-credential granted.*

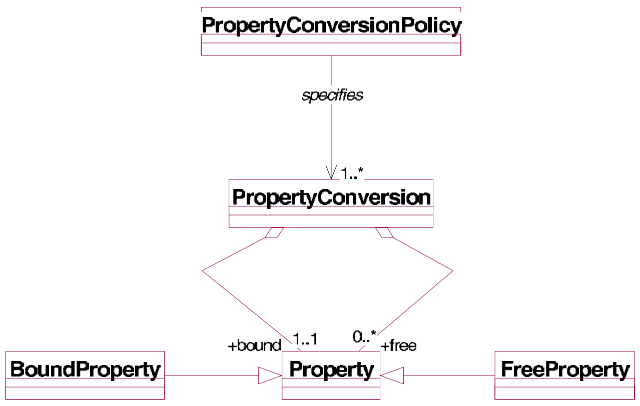


Figure 4.10.: A class model for the conversion of free properties into bound properties

The middle part of Figure 4.9 visualizes the situation. The entity on the right is the grantor following a property conversion policy. The entity in the center requests a promise for a permission, i.e., a bound property. The grantor:

- verifies the submitted free property-certificates with the supporting licences,
- extracts the contents of the free property-certificates and interpretes them as free properties,
- applies his conversion policy on the extracted free properties, and
- finally, if all checks have been successfully completed, grants a bound property-credential where the subject (grantee) is the same as in the submitted free property-certificates.

Figure 4.10 visualizes a class model for the structure of the property conversion policy in regard to converting free properties into bound properties.

4.5.2. Composing Certificate Processing, Credential Processing and Their Links

An instance of the full hybrid PKI model consists of overlapping components of three kinds:

- trusted authorities (also called trustees) and licensees for and a holder of a free property (see Section 4.3) together with a verifier of this free property,
- an owner and delegatees for and a grantee of a bound property (see Section 4.4), and
- a holder of free properties and a grantor of a bound property (see Section 4.5.1).

Components of the first two kinds can form so-called loops. A verifier for a free property in a component of a first kind *has to close* the verification loops to the trusted authorities in order to found his trust. More precisely, closing the loop means here that the verifier *decides* which entities he wants to trust and to accept as origins. A grantee of a bound property *automatically closes* a loop when addressing the pertinent owner. A component of the third kind is used as a link between a component of a first kind and a component of the second kind. The link identifies the verifier of the former component with the grantor of latter component, and the holder of the first component with the grantee of the latter component. In some special cases, a linked component may appear to be degenerated. This also allows to subsume traditional access rights management under the hybrid model. The links are the important feature that makes the hybrid model highly flexible and adaptable for many applications. Figure 4.9 shows a simple example in which two loops for chains are linked.

4.5.3. Classification of Entities

In the following we classify entities involved in an instance of the hybrid model (see Figure 4.9 in Section 4.5). Figure 4.11 visualizes a class model formalizing this classification. The class model also represents the entities involved in any instantiation of the model of trusted authorities and licencing as well as of the model of owners and delegation. This class model can be understood as an extension of the class *entity* which is illustrated on the right side of the lower part of Figure 4.1.

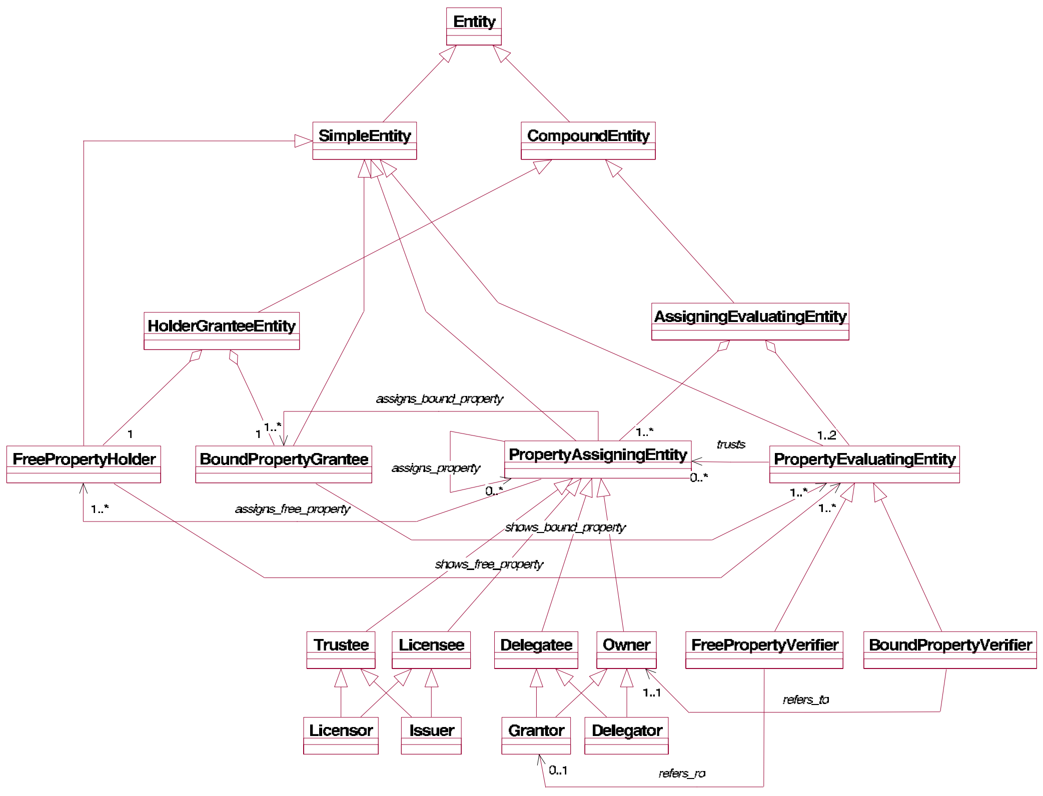


Figure 4.11.: Entities involved in an instance of an hybrid PKI model

The classification of the entities comprises two kinds of entities: *simple entity* and *compound entity*.

A *simple entity* might be a *free property-holder*, a *bound property-grantee*, a *property assigning-entity*, or a *property evaluating-entity*.

A *property assigning-entity* might be a *trustee*, a *licensee*, a *delegatee*, or an *owner*. Each of these entities may assign administrative or characterizing properties to other entities. A *trustee* as well as a *licensee* might act in the role of a *licensor* or an *issuer*. Both a *delegatee* and an *owner* might act in the role of a *grantor* or a *delegator*.

A *property evaluating-entity* might be a *free property-verifier* or a *bound property-verifier*. An entity as a free property-verifier may refer to a *grantor*, i.e., in an instance of the hybrid PKI model. An entity as a bound property-verifier refers to an entity as owner.

Compound entities occur both in the model of owners and delegation and in the hybrid PKI model. A *compound entity* acts in two roles in any instantiation of each of these models. A compound entity might be a *holder/grantee-entity* or an *assigning/evaluating-entity*. A holder/grantee-entity consists of a *free property-holder* and a *bound property-grantee*. A holder/grantee-entity occurs in an instance of the hybrid PKI model (see Section 4.5.2). An assigning/evaluating-entity occurs in any instantiation of the model of owners and delegation as well as of the hybrid PKI model.

In the model of owners and delegation, an assigning/evaluating-entity comprises either one[16] property assigning-entity (an owner as grantor) or at least two[17] property assigning-entities (an owner as delegator, delegatees as dependants, and a delegatee as grantor). The same assigning/evaluating-entity consists of one property evaluating-entity which is a bound property-verifier that refers to an owner. In the hybrid PKI model, this assigning/evaluating-entity comprises a free property-verifier (an entity as grantor) and a bound property-verifier (an entity as owner). In the simplest case where no delegation is used, both property evaluating-entities refer to the same entity as owner.

4.6. Evaluation of the Hybrid Model Against the Core Concepts of Secure Mediation

In Section 3.4, we have presented the main characteristics of a security framework which should be a basis for developing the approaches of i-mediation and f-mediation (see Chapter 3). We argue that the hybrid PKI model presented in this chapter constitutes a kind of security framework which establishes a significant basis, on which the pure approaches of i-mediation and f-mediation, as well as their combinations as stated in the beginning of this chapter, can be implemented.

In the following we refer to the main characteristics of the security framework and show the concepts of the hybrid model matching the main characteristics which have been introduced. For the sake of perspicuous exposition, we review these characteristics.

I. An entity is represented by (one of) its public keys and characterized by the

[16]This situation occurs in the simplest case (see Section 4.4.1).
[17]This situation occurs in more advanced cases where delegation is used (see Section 4.4.1).

assigned properties.

This basic characteristic is treated by the concepts presented in Sections 4.2.1 and 4.2.2.

II. An entity follows a security policy expressed in terms of characterizing properties.

The basic characteristic II is treated by the concepts presented in Section 4.2.4.

III. An entity E affirms that entity F has a certain property by assigning the properties of entity F to entity F's public keys.

This main characteristic is handled by the concepts presented in Section 4.3.1.

IV. An entity E trusts, on its own discretion, entity F's judgements on a property.

This main characteristic is also handled by the concepts presented in Section 4.3.1.

V. An entity E delegates the authority over evaluating a property to entity F, i.e., entity E trusts entity F's judgements on the property.

The main characteristic V is served by the concepts presented in Section 4.4.1.

VI. An entity uses one property to make inferences about another property.

Finally, Section 4.5.1 presents concepts dealing with the main characteristic VI.

4.7. Contributions

As seen above, previous approaches for defining a PKI are classified as based either on trusted authorities with licencing or on owners with delegations. In this chapter we identified the similarities and the differences of these approaches. Roughly summarized, they are similar with respect to the need of supporting documents for a main document, a certificate or a credential, respectively, but they differ with respect to how trust evaluations are performed. We argue that many applications require to use and to link both kinds of PKI. Accordingly, we presented a hybrid PKI model which unifies and extends the previous approaches.

More precisely, we made following main contributions:

- We first presented property assignment and trust evaluation, which are two basic functionalities of any PKI, in a generic and unified conceptual model. It is generic, since two prevalent PKI models can be instantiated from this conceptual model. It is unifiying, since previous approaches uses different concepts and terms, some of which, are overloaded. Our conceptual model gives a unified view on existing models.

- We introduced explicit *administrative properties* to be encoded in supporting documents.

- We also introduced *bound authorization attributes* which are more general than capabilities.

- We then exhibited the *generic structure of links* between instances of the previous PKI models. Basically, in such a link an entity employs its security policy in order to convert submitted certificates assuring *free properties* of the holder of a public key into *bound properties* promising the holder to permit access to some service.

- Afterwards, we presented a comprehensive class model of entities. This model represents all entities involved in two prevalent PKI models and a hybrid PKI model.

- Finally, we have evaluated the hybrid model against the core concepts of the aimed security framework.

As a summarizing contribution, we presented a flexible hybrid PKI model which establishes a significant basis of trust management needed for a broad spectrum of applications.

5. Applications of the Hybrid PKI Model

Though the two underlying models presented in Section 4.3 and Section 4.4 are quite different in their detailed requirements and their purpose for the hybrid model, there may arise situations which allow the use of either of them, however notably with different semantics. To demonstrate its flexibility, we present two practical application scenarios as instantiations of the hybrid model.

5.1. Finding the Proper Service Provider

In this application we consider the following situation: An entity tries to determine which other entities are willing to offer him a specific service; he then requests that service.

In a first version, the service seeking entity collects suitable free property-certificates and supporting licence-certificates and then sends a corresponding *dag* (see Section 4.2.3) to some mediating entity (e.g. a f-mediator in the sense of Section 3.3) together with the request for a bound property-credential for the wanted service. The mediating entity decides on the request according to his property conversion policy and based on the delegation-credentials which have previously been granted to him by appropriate providers. In the positive case, the mediating entity selects a provider and grants a suitable bound property-credential to the service seeking entity together with the supporting delegation-credentials, which refer to the selected provider as a unique origin. After receiving the corresponding dag, the service seeking entity can address the provider and show his eligibility for the service by showing the received dag.

In another version, the service seeking entity acts as an owner of his personal special mailbox for offered services. In principle, he can grant a bound property-credential as a capability that allows other entities to insert an offer in his personal special mailbox. Rather than granting such a credential by himself (since he does not know to whom to grant it), he grants a delegation-credential for the capability to some mediating entity. If the mediating entity knows an appropriate provider, the mediator can grant the capability to the provider and forward the delegation-credential. Assuming that the provider received such a credential, he inserts his offer by using the corresponding chain. If the service seeking client is willing to accept the offer, he collects suitable free property-certificates and supporting licence-certificates and then sends a corresponding dag to the provider together with a request for the wanted service. Finally, the provider decides on the request according to his security policy. In the positive case, he immediately executes the service being seeked. Surely, some further modifications are possible. The crucial point is to observe that the same high level functionality, *seeking a provider and requesting its service*, can be refined in different ways using the flexibility of the hybrid PKI model.

5.2. Secure Mediation of Roles

In the following we present a role-based secure mediation approach by combining the strengths of role-based access control [79, 36] and hybrid PKI model.

According to the approach of role-based access control (RBAC), permissions are associated with roles, and entities are granted membership in appropriate roles, thereby acquiring the roles' permissions. Access to a service requested by an entity is granted, if the requesting entity enjoys necessary roles associated with the requested service.

In the pure design of role-based access control, roles are assigned to users based on the administratively-defined identities. Furthermore, the role membership is controlled by a few centralized authorities of a single organization, since RBAC was developed for access control in a single organization.

In contrast to RBAC, in a role-based secure i-mediation where an i-mediator and each data source are supposed to follow role-based security policies, each resource owner would grant membership in roles based on the characterizing properties of the clients which are unknown prior to their requests. Membership in a role is supposed to be captured by a digital document. Roles are treated as special bound authorization attributes (see Section 4.2.2). Thus, granting role membership to an entity means granting an appropriate bound property-credential to the same entity within the domain where the role is defined. Outside a domain, such bound property-credentials may be interpreted as free property-certificates by other resource owners, i.e., who interpret the bound properties of other domains as free properties and use them as grantees in their security policies.

In such a role-based secure *i*-mediation, it is quite possible that a resource owner may follow a role membership policy specifying

> *which set of personal attributes a client has to enjoy, in order to obtain a local role assignment.*

In another role-based *i*-mediation, a resource owner who is willing to serve a well-directed group of clients, i.e., those which are the registered users of potential partner-organizations of the resource owner, may follow a role membership policy specifying

> *which set of personal attributes and which set of partner organization-roles a client has to enjoy, in order to obtain a specific local role assignment.*

It is important to note that a *partner organization-role* is used as a bound property within the organization in which the role is defined. Another resource owner outside the organization may interpret this bound property as a free property.

Independently of a specific role membership policy, the accomplishment of the implementation of any role membership policy for *i*-mediation is affected whether or not the participants of *i*-mediation share a common security domain:

- In case of existence of a common security domain, including common vocabularies for personal attributes and role names, role membership is controlled by the respective autonomous resource owner. Expressed in terms of properties, this means that a role membership is granted to a client, represented by (one of) its public keys, by the respective resource owner after evaluating the shown characterizing properties – those which are the personal attributes or partner organization-roles – assigned by some trusted authorities or granted by the trusted owners of the partner organizations, respectively.

  In a specific *i*-mediation scenario, where an *i*-mediator is placed as a filter in front of data sources, the *i*-mediator may expect that a source will grant appropriate role memberships to a client and may forward the client's request including the client's suitable asserted characterizing properties to the source. This expectation could be based on the representation of the source's policy, which might have been known by the i-mediator prior to the client's request. The i-mediator's decision on forwarding the client's request can be seen as a part of i-mediator's secure mediation policy. However, each data source autonomously decides for granting appropriate role memberships based on the received requests.

- If there exists no common security domain shared by the participants of the i-mediation, participating entities are confronted by a common challenge arising from the heterogeneity of the different, but logically related security domains, in which these entities are hosted. This heterogeneity may refer to different vocabularies used for personal attributes and role names and different formats used for certificates. Due to the syntactical and semantic heterogeneity which could arise, the entities cannot interoperate with each other without homogenizing these differences in some way.

A way of establishing interoperability between the entities of two heterogeneous and autonomous security domains is to build coalitions between these security domains. A coalition between two security domains requires a coalition agreement aiming at building common vocabularies and a contractually involved cross-certification [88, 53] aiming at connecting trusted authorities of both security domains. Unfortunately, this solution hides some challenging issues as outlined in the following:

- In principle, two partners of a coalition can agree on common vocabularies with respect to personal attributes or role names. However, a coalition agreement process cannot be fully automated due to the semantic differences regarding vocabularies. A coalition agreement requires costly human intervention, and thus noticeable loss in time.

- Cross-certification is a way of establishing interoperability between two PKI hierarchies under cross-certification agreements. The top most trusted authority of a security domain A signs a certificate vouching the public key of the top most trusted authority of the security domain B and vice versa. Cross-certification would make PKI hierarchies truely open, but true interoperability goes beyond inter-hierarchical communication. The main problem opposing this trend, is that trusted authorities acting in different security domains generally have incompatible certification policies, whereas cross-certification requires some sort of certification policy alignment. Governments can, for example, enforce a common certification policy and cross-certification between all PKIs used by the different government units, but ad-hoc cross-certification between commercial and organizational PKIs is difficult to achieve due to heterogeneous certification policies. Cross-certification requires that the certification policies are to be equal strength in involved PKIs. However, policy strength is a multi-dimensional measure that is difficult to match. Another challenge refers to the bits in the certificates. Cross-certification needs interoperable certificates, each of which is valid across organizational boundaries. For this to occur, PKI vendors need to agree on standards that allow certificates to be implemented across secured applications. CertifIcates are unfortunately not the same across the board. There are a number of reasons why a verifying entity may not be able to decode a certificate. Some of them are: complexity of the developed standards, unsupported algorithm which is to be used to sign a certificate and unsupported certificate version.

We argue that the main challenge lies in the policy issues surrounding certificates.

Based on these challenges, with which seem hard to cope, and on the dynamics of i-mediation (i.e. with respect to spontaneous users and ad-hoc collaborations), we argue that such coalitions as discussed above are not easy to build and, in particular, not useful for i-mediation. Even if all technical challenges can be mastered, a problem remains unsolved. A resource owner, who is willing to serve a wide range of potential clients, will not always know most of its potential clients and thus the associated security domains.

To handle the decentralized nature and the dynamics of role-based i-mediation, we employ the approach of f-mediation (see Section 3.3). Using f-mediation, a resource owner is able to transfer its authority of granting role memberships to other trusted entities which have the required domain expertise as well as previously established coalitions. A trusted entity enjoying such an authority, with respect to granting role memberships, can then grant appropriate role memberships to potential clients based on the clients' personal attributes and clients' roles which have been assigned and granted to the clients in their security domains. These domains are in general incompatible with the security domain of the resource owner.

In the following we give an example showing how we combine the strengths of RBAC and hybrid PKI model to demonstrate secure distribution of roles across organizational boundaries.

We consider an i-mediation scenario where autonomously operating forensic institutions offer anonymous forensic data about sex offenders and a fictitious i-mediator conducted by the European Community is specialized in European forensic institutions. Besides serving spontaneous users, the i-mediator may also want to share some part of its data only with a privileged group of users, e.g., the researchers working in US forensic institutions. The main motivation of the i-mediator is to broaden its potential collaborators basis. For this purpose, the i-mediator following a mediation specific RBAC policy may define

- some *free-roles* (e.g. law student), memberships of which should be granted only on the basis of shown personal attributes (e.g. student identification from the school of law) of the spontaneous users (e.g. law students), and
- some *privileged-roles* (e.g. visiting researcher), memberships of which should be granted to the privileged users (e.g. researchers working in forensic institutions) on the basis of a user's personal attributes in combination with the evidences of her eligibility (e.g. proof of a role membership[1]) in terms of her organizational responsibilities and activities.

The i-mediator may follow a security policy specifying

> *An entity can be granted the local role "visiting researcher", if she is a "psychoanalyst" and working in a US forensic institute as a "principal investigator".*

It is important to note that the expected role "principal investigator" of a potential client could be a *role* which has been granted to the client according to the RBAC policy of the client's institution. This expected role could also be a *group*

[1]Such a role membership is encoded in client's bound property-credential which is interpreted as a free property-certificate outside her institution.

membership which has been granted to the client by her institution without knowing the later usage of this membership outside the organization. In both cases, the expected role "principal investigator" is treated as a bound property within the client's institution and the owner of the institution is the grantor of an institution specific bound property. From the i-mediator's point of view, the asserted property "principal investigator" is interpreted as a free property.

We can hardly assume that the i-mediator, as well as the forensic institutions abroad, can reach a common understanding about personal attributes and roles implicitly. To reach its potential clients, the i-mediator transfers its authority of granting membership regarding the role "visiting researcher" to an f-mediator, e.g., the head of a corresponding FBI unit having the required expertise about mapping between the properties used in i-mediator's security policy and the properties used in the security policies of US forensic institutions. For this purpose the i-mediator as a resource owner grants a delegation-credential encoding the role "visiting researcher" and eventually the corresponding security policy specifying the role membership to the head of the FBI unit.

A potential client working in a US forensic institute collects her accreditation-certificates and the supporting licence-certificates. She also collects her bound property-credentials[2] and associated delegation-credentials which refer to the head (owner) of the institution for which she works. She sends the corresponding dags to the FBI unit with the request for a bound property-credential for the wanted privileged-roles. The FBI unit decides on the request according to its security policy and based on the delegation-credential which has been granted to it. The FBI unit as the grantor of a bound property credential encoding a privileged-role:

- verifies the submitted accreditation-certificates with the supporting licences,
- extracts the contents of the accreditation-certificates and interpret them as personal attributes,
- verifies the submitted bound property-credentials with the supporting delegations,

[2]Seen from the viewpoint of the FBI unit, these credentials are interpreted as free property-certificates.

- extracts the contents of the bound property-credentials and interpret them as free properties (e.g. personal attributes),
- finally, if all checks have been successfully completed, grants a bound property-credential where the subject (grantee) is the same as in the submitted accreditation-certificates and bound property-credentials.

The evidence of eligibility for the client's expected personal attribute "psychoanalyst" is supposed to be encoded in a submitted accreditation-certificate issued by an accredited university.

It is important to observe that, in addition to verifying personal attributes, the FBI unit as a grantor of a bound property-credential also verifies the client's bound property-credentials with supporting delegations. This requires the FBI unit to treat the head (i.e. which is the origin of the corresponding dag) of the institution, for which the client works, as a trusted authority and to accept bound property-credentials granted by this owner. The trust relationships between FBI and corresponding forensic institutions are supposed to be regulated by contractually involved cross-certifications.

Surely, some modifications are possible. The crucial point is to observe that the same high level functionality, *assured distribution of roles across organizational boundaries*, can be refined in different ways by combining the strengths of RBAC and our flexible hybrid PKI model.

Traditional access rights management with capabilites [43] or with access control lists [45] can be seen as degenerated special cases which exploit identifying names as free property. In both versions, an identifying name as a free property is connected with the permission to access a service as a bound property. With traditional capabilities, the bound property is explicitly delivered to the client. With access control lists, the connection remains under the control of the server. Accordingly, granting capabilities requires to authenticate the recipient and to take precaution against stolen and subsequently misused capabilities. Using access control lists requires to identify and to authenticate a requesting client.

In an *i*-mediation scenario with a common security domain shared by the mediation participants, the role membership can be directly controlled by the respective autonomous resource owner. This situation constitutes a special case where bound properties – in this case roles – are not explicitly delivered, since the corresponding role memberships are granted directly by the resource owners, but not by the distributed authorities. In such scenarios, f-mediation can still be employed for other goals, e.g. to distribute the management load with respect to granting role memberships.

In another i-mediation scenario with no common security domain shared by the mediation participants, the role membership is controlled by the distributed authorities. In this situation, the bound properties – roles – are explicitly delivered. In this scenario, before a resource owner decides on a permission for his service based on a shown bound property-credential, the resource owner is assumed to follow good practice in order to authenticate the subject, i.e., a public key. Typically, the client has to correctly respond to a challenge generated by the resource owner.

5.3. Contributions

In this chapter we presented some applications of the hybrid PKI model presented in Chapter 4. We made the following contributions. First, we demonstrated the applicability and flexibility of the hybrid model by choosing two practical application scenarios. Second, seen from the viewpoint of the role-based access control research efforts, we exhibited how distributed RBAC can be implemented for dynamic coalition environments by employing our hybrid PKI model. This shows that our solution appears to be worthwhile for a broad spectrum of applications.

Part IV.

Secure *I*-Mediation

6. A Specific Secure *I*-Mediation

6.1. Introduction

In Section 3.2, we have discussed the motivations for employing *i*-mediators and *i*-mediation. The *i*-mediation paradigm needs powerful and expressive security mechanisms considering the dynamics and conflicting interests of *i*-mediation participants. In this chapter we present the security requirements on *i*-mediation, the general design of secure *i*-mediation, and the specific *Multimedia Mediator*, MMM, security architecture.

We discuss the security requirements for *i*-mediation with an emphasis on *confidentiality* and *authenticity*. The main insights are the following:

- In general, there is no a priori organizational base for direct mutual trust between users, *i*-mediators and sources.

- Confidentiality and authenticity should be based on *certified personal authorization attributes* which express a user's eligibility to see a certain kind of information. Additionally, free properties that identify the user can be used, but in many situations identification of a user is neither needed nor appropriate for *i*-mediation. Avoiding identification is also a necessary condition for anonymity, and thus it enables the use of dedicated anonymity techniques.

- Information sources should decide *autonomously* how they interprete submitted personal authorization attributes, i.e., which attributes they require in order to return some information.

- *I*-Mediators should be able to *forward* submitted personal authorization attributes, and to *assist* users to gather appropriate ones for their information needs.

In order to meet these requirements we take the advantage of the PKI model of trusted authorities and licencing (see Section 4.3). Based on the main requirements presented above and the underlying PKI model we present a general design for secure i-mediation. Shortly summarized, secure i-mediation is outlined as follows:

- A user submits evidence of being eligible for seeing the answer to a query by submitting certified *personal authorization attributes* which are encoded in *accreditation-certificates*[1].
- An i-mediator examines, selects and forwards submitted accreditation-certificates together with appropriate subqueries to the data sources.
- A data source autonomously bases *access decisions* for requested data on shown accreditation-certificates, and it returns subquery answers in *encrypted form* to the i-mediator. For encryption a data source applies the user's *public key* which is also contained in the accreditation-certificates for an asymmetric encryption scheme.

  In this point it is important to note that the model of trusted authorities and licencing is used slightly different in the context of secure i-mediation. In this PKI model as well as in other PKI models discussed in Chapter 4, the verifiers employ appropriate challenge-response protocols in order to authenticate the public keys included in the certificates and credentials shown by the requestors. As a special case, in the specific secure i-mediation, a data source does not need to employ a challenge-response protocol to gain assurance whether the holder (requestor) of a shown asserted free property is also the holder of the matching private key, since each data source encrypts its subquery answers with the user's public key and only the unique owner of the accreditation-certificates can recover the plaintext and thus gain the requested information.
- Then, the i-mediator processes the returned encrypted data in order to produce the final, still encrypted query answer.

So far, the identified requirements and the proposed design for secure i-mediation apply to a wide range of i-mediation technologies. In order to implement the gen-

[1]In [7], we used the term *credential*. In this chapter we would prefer to call the document an *accreditation-certificate*. However, the document is used like a *credential* in the sense of [7].

eral design, the specific approach to i-mediation has to be taken into account. For this purpose, we present the *security architecture* for the prototype of the *Multimedia Mediator*, MMM, [14, 15]. The MMM constitutes a specific i-mediator, basic functionality of which is presented in Section 3.2.

The MMM is intended to provide an application oriented view on potential sources. Application orientation is achieved by an application schema which is supposed to be designed and declared by an i-mediator administrator according to the information needs of the foreseen clients. Formally, the application schema is just an object-oriented database schema, like in traditional database technology. However, in contrast to traditional technology, no data instance is stored. Rather, on receiving a client's query, the MMM dynamically extracts the data needed for the answer from the connected sources. Such data extraction is based on (partial) embeddings of the application schema into the schemas of the sources (if they exist or into other suitable self-descriptions).

The rest of this chapter is organized as follows: In Section 6.2, Requirements, we discuss the needs of secure i-mediation, thereby emphasizing the differences and the new challenges in comparison to the more traditional federated approach [47, 82, 63]. In Section 6.3, Design, we present the general design for secure i-mediation, and we examine its security properties. In Section 6.4, Architecture, we specify the details of the prototype implementation for our specific approach to i-mediation.

6.2. Requirements

We anticipate that the information trading market will expand in the future and that information integration will become an indispensable service for many clients. When transferring observations of real world markets to information system models, especially dynamics and conflicting interests are to be taken into account.

Clients demand systems enabling them to effectively work with heterogeneous information sources. This demand stimulates information sources to supply their information on an ad-hoc basis, in particular for purchase. Using i-mediators, clients can find and correlate information more efficiently. Information sources are likely to meet the client's requirements and to cooperate with i-mediators. Like in a marketplace of supply and demand, there exist different motivations for cooperation between the participants in i-mediation. Generally clients and information sources are inde-

	federated approach	***i-mediation approach***
interoperability stimulated	by members	by spontaneous clients
relationship between participants	organizational	contractual
trust between participants	high	low
anonymity need of clients	low	high
authorization policies	based on identities	based on character-izing properties
design	bottom-up	top-down
set of information sources	static	dynamic
union of source data	= all relevant data (closed world)	$\subseteq$ all relevant data (open world)
data at the integration layer	virtual	hybrid
client registration	static	spontaneous & dynamic

Table 6.1.: Federated vs. *i*-mediation approach

pendent of *i*-mediators. Clients are independent of information sources if there are several suppliers for their demand. Information sources exist in competitive as well as in non-competitive relationships with other information sources. Obviously there is no a priori base for direct mutual trust between the participants in *i*-mediation.

While information sources may have cooperation contracts with *i*-mediators, it can be assumed that spontaneous clients are unknown beforehand. Clients thus cannot be registered with *i*-mediators in a static way before queries can be satisfied. Even the group of information sources probably will not be as stable as found in federated systems due to the lack of organizational associations in *i*-mediation. Though one or more information sources may be temporarily unavailable a client query can be satisfied. There apparently is no useful assumption of a closed world in *i*-mediation due to their dynamics.

Table 6.1 summarizes some of those key differences between federated and *i*-mediation approaches, which also affect security considerations. These differences, as well as the similarities, of the two approaches are visualized by Figure 6.1 and Figure 6.2 and further explained as follows.

In contrast to *i*-mediation, the federation establishment is stimulated by the members of the information source organization in order to support a closed group of registered clients. In many cases, the client group is part of the organization which supports its interactions by means of the federation. Furthermore, there exist dependencies between clients and information sources due to their identical organizational origin. The information sources of a federation act in a common interest anchored in the organization they belong to. This network of dependencies has probably had significant impact on specific security architectures, as often found in federated database systems [54, 85, 28].

I-Mediation is more suitable to model dynamics and low trust between interacting parties. An *i*-mediator's top-down design paradigm (see Figure 6.2) allows

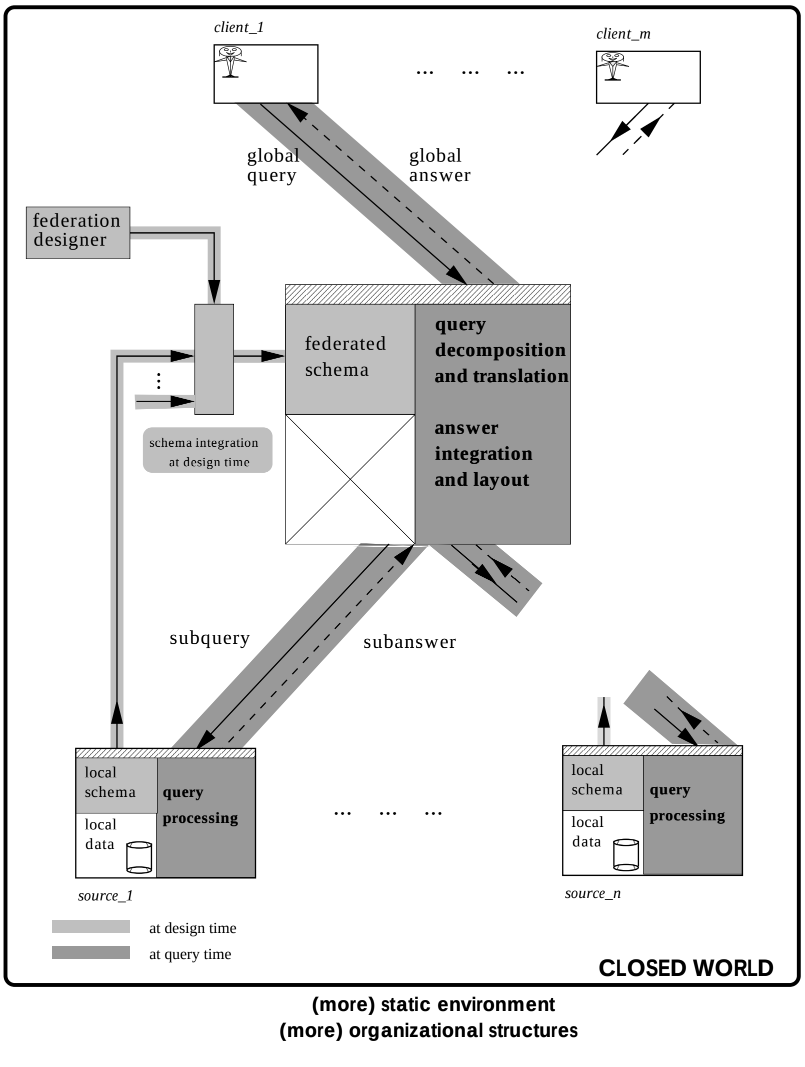


Figure 6.1.: Design of federated systems

for a stable data description of integrated information at varying degrees of source fluctuation, whereas a bottom-up approach (see Figure 6.1) requires a redesign of the global integration schema each time a local schema changes or is added. *I*-Mediators strive for tolerance with respect to information provider failures and offer services to ad-hoc clients which have not registered with the services beforehand.

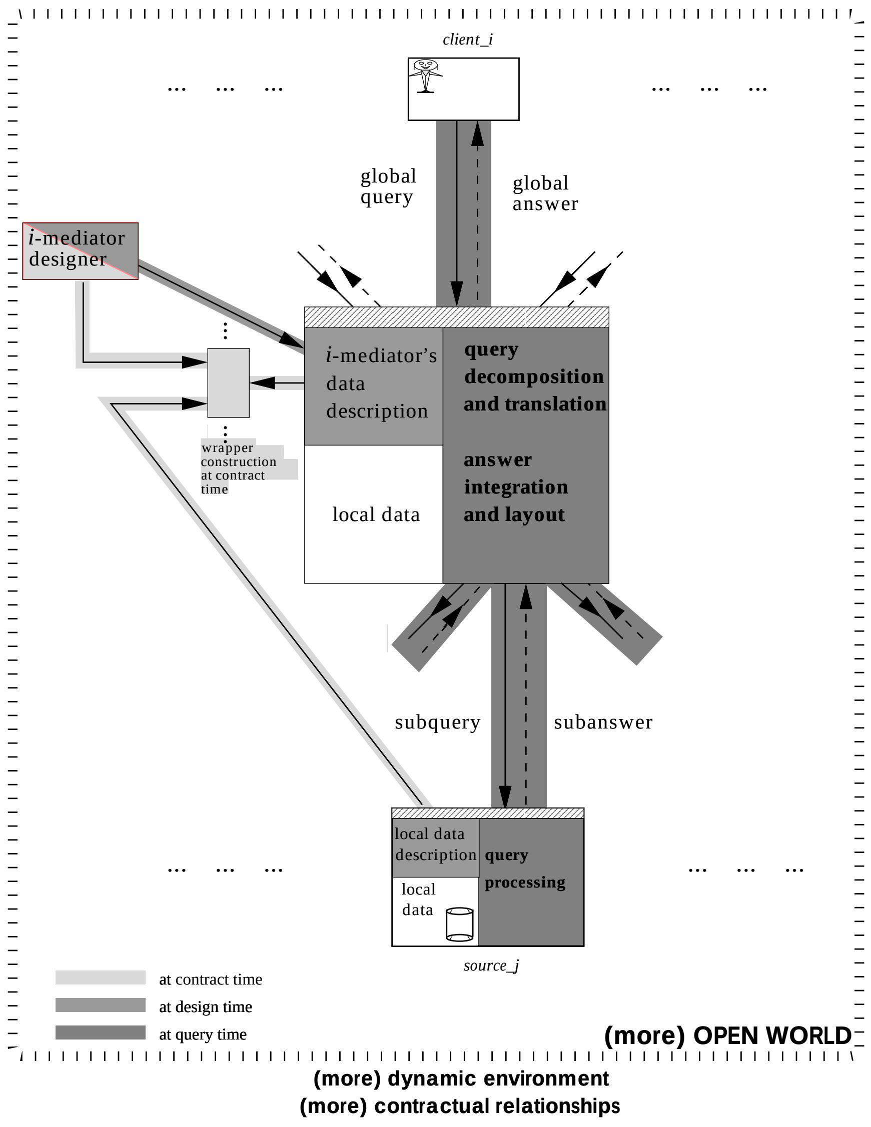


Figure 6.2.: Design of *i*-mediation

Due to this spontaneity, the employment of merely identity-based authentication and authorization approaches, as traditionally used in federated systems, is less useful for i-mediation. Given that the identities of clients cannot be known in advance to the receivers of the requests, these identities cannot be put on the access control lists (ACL) and other identity-based authorizations of the receivers to decide who can do what.

Based on the above, one can ask if identity-based approaches are really a good idea in this context. The clients may be unwilling to divulge their identities for private reasons and thus prefer to remain anonymous. Additionally for a resource owner, it may be necessary to see evidences of a client's eligibility (e.g. profession, organizational membership, security clearance, academic title) rather than to know *who* they are.

We consider the following example scenario motivating our secure i-mediation approach. A psychoanalyst working in a project of a university's school of law wants to know whether there exists a correlation between watching a certain kind of film and committing crimes of violence. For this purpose, she may need to send at least the following query and evidences of her eligibility to an i-mediator specialized in forensic and movie sources.

QUERY. *Find the motive and offence for all offenders which have seen films with a violent content.*

In order to answer this query the i-mediator may need to query a forensic source and a few independent movie sources, each of which may impose various types of security restrictions. While a forensic source may need to know whether the client is a member of an organization with which the forensic source cooperates, a movie source may be interested in knowing whether the client is working for a UN country.

6.3. Design

Based on the requirements discussed in Section 6.2, we present a general design for secure i-mediation in the following sections.

6.3.1. Fundamental Security Requirements for Querying

The following fundamental security requirements for querying are considered:

- Any source wishes or is even legally obliged to autonomously follow a security policy, with respect to confidentiality, which ensures that requested information is delivered only to appropriate clients. In order to achieve this goal, clients have to provide evidence that they are eligible for requested information, and sources have to maintain mechanisms to inspect such evidence and to decide whether and which information is returned. Furthermore, a source has to ensure that information is actually delivered to only that client which provided the inspected evidence.

- The policy, with respect to confidentiality as stated above, should be at least compatible with additional viewpoints concerning authenticity, anonymity, integrity and availability.

- Any client wishes that shown evidences cannot be misused.

Surely, these requirements should be met for the simple case in which a client directly addresses a source, as well as when both the client's request and the source's delivery are mediated.

6.3.2. Basic Informational Environment

Based on the model of trusted authorities and licencing (see Section 4.3), we assume that there are trusted authorities, trusted by all participants of a transaction. A trusted authority affirms that a participant has a certain property by assigning the properties of the participant to participant's public keys. A trusted authority offers at least following services:

- A trusted authority signs an *identity certificate* of the rough form

  ⟨identity(address), public encryption key, public verification key⟩,

  thereby assuring that the *participant* specified by the first component is the legitimate owner of the matching private keys for decryption and signing.

- A trusted authority signs an *accreditation-certificate* of the rough form

  ⟨personal attribute, public encryption key, public verification key⟩, thereby assuring that the *personal attribute* specified by the first component is enjoyed by the legitimate owner of the matching private keys for decryption and signing.

As noted in Section 6.1, since each data source encrypts its subquery answers with the user's public key included in the shown certificates, data sources do not need to employ a challenge-response protocol in the model of trusted authorities and licencing as instantiated for secure i-mediation.

In a secure i-mediation, the personal attributes appear in the role of *personal authorization attributes*. A client is respresented by (one of) its public keys and characterized by the assigned personal authorization attributes. A source follows a security policy in terms of personal authorization attributes and trusts, on its own discretion, a trusted authority's judgements on certain personal authorization attributes.

For our design of secure i-mediation we assume that the participants agree on a *common understanding* of personal authorization attributes, certificate formats, and the certification policies under which the certificates are issued. We also assume that the personal authorization attributes of a client are certified for a properly authentificated key.

Our basic protocols employ the personal attributes as evidence whether the owner of the matching private keys is eligible for some requested information, thus deciding whether and which information is returned. It is important to observe that the source does not care how it has obtained knowledge of the accreditation-certificates, whether directly from the owner of the matching private decryption key or otherwise.

For the basic protocols we only need the *public encryption keys*, as sketched in the following. Suppose that a participant wants to ensure that some returned data contain meaningful information only for the supposed owner of the matching private decryption keys. Then the participant takes care that the delivered data is the ciphertext of the plaintext which contains the information under consideration, where the encryption is done with the public encryption keys. The public verification keys are merely provided as a precaution for protocol extensions.

6.3.3. Secure Direct Querying

Given this basic informational environment, we can specify the basic protocols. We distinguish a preparatory phase and a query phase.

In the *preparatory phase*, clients and sources do not yet interact. A *client*, wishing to request information later on, collects accreditation-certificates with her personal authorization attributes. And a source, entitled to answer queries later on, defines a *security policy* with respect to confidentiality which relates personal authorization attributes to the amounts of information allowed for delivery. More precisely, a security policy is abstracted to be specified in the following form:

- As input, the policy accepts some set of accreditation-certificates belonging to a unique owner or at least a group of clients which consciously cooperate. It is important to observe that it is not necessary for the source to know the identity of that owner.
- Only based on the personal authorization attributes shown by the accreditation-certificates, the policy states which kind of information is allowed to be delivered to the owner.

In the *query phase*, the **protocol for secure query answering** is applied (see Figure 6.3). It has two subphases, a request phase and a delivery phase. The protocol is outlined as follows.

Request phase

1 The client sends a request ⟨return information, query, set of accreditation-certificates⟩ to the source.

Delivery phase

2 The source verifies each accreditation-certificate and determines the associated personal authorization attributes.

3 The source evaluates the query under the restriction that only such information is generated that is allowed to be delivered on the basis of the associated personal authorization attributes.

4 The result of the restricted query evaluation is considered as plaintext and encrypted with an appropriate set of public keys occuring in the set of accreditation-certificates. Instead of using the asymmetric encryption scheme for encrypting the result the source can exploit a hybrid scheme, i.e., encrypt the answer with a session key using any encryption method, and then encrypt only the session key with the public keys.

5 The resulting ciphertext (and the encrypted session key, if applicable) is sent back following the directions given by the return information.

This protocol with appropriate and subsequently described extensions satisfies the following security properties while complying with the fundamental security requirements, as stated in Section 6.3.1:

- Clients and sources exclusively have to trust the trusted authorities that signed the accreditation-certificates.
- Sources are supposed to have an interest in checking eligibility. Thus, for instance, data subjects the data of which is stored in a source have to trust the source with respect to checking eligibility appropriately.

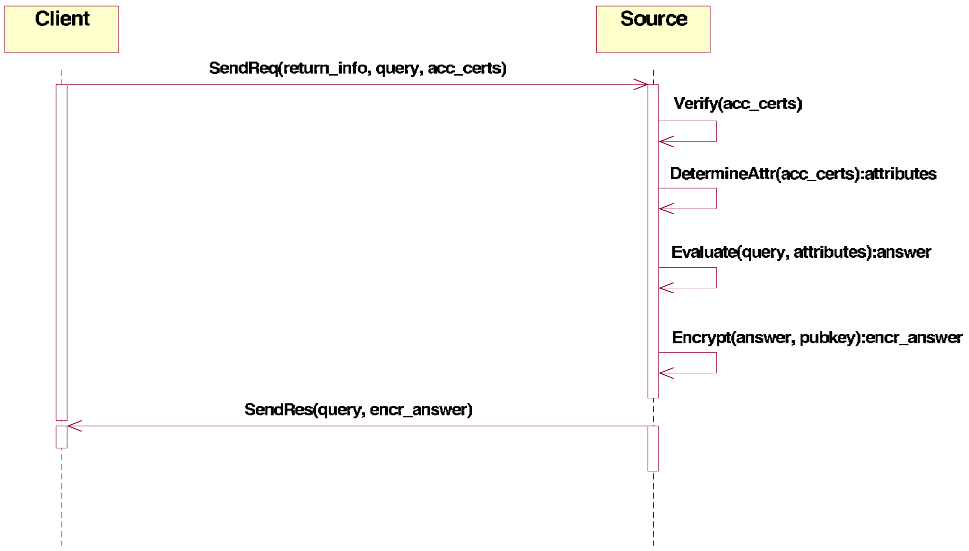

Client
Source
SendReq(return_info, query, acc_certs)
Verify(acc_certs)
DetermineAttr(acc_certs):attributes
Evaluate(query, attributes):answer
Encrypt(answer, pubkey):encr_answer
SendRes(query, encr_answer)

Figure 6.3.: Protocol for secure direct query answering

- Eligibility is supposed to be definable in terms of personal authorization attributes.

- Since personal authorization attributes are shown in the form of accreditation-certificates that do not contain a field for the identity of the owner, a client can stay anonymous as far as the source cannot infer the identity from its knowledge about the personal authorization attributes, from the communication protocol data and the query, as well as the answer.

- Since a source cannot always effectively decide whether two given accreditation-certificates belong to the same person, certain kinds of security policies need special treatment. In particular, the constraint that only one person shall be authorized to receive results requires that the submitted accreditation-certificates are linkable, in the simplest case by just containing the same keys. Otherwise, a group of persons may collude to show eligibility for some information that no single group member is allowed to see. In contrast, the four-eyes principle would demand to ensure that the submitted accreditation-certificates are definitely not linked.

- Even in an untrusted network, only the unique owner of the verified accreditation-certificates can recover the plaintext and thus gain the requested information. However, in some situations, we have to take care of possible plaintext attacks aimed at discovering the corresponding private keys. These situations are given if an attacker is herself eligible for another client's eligible request and we do not use a hybrid encryption scheme. A possible countermeasure would be to employ nondeterministic encryption, i.e., adding some random data to the plaintext before encrypting it.

- If an attacker knows a client's query and is herself eligible for this query, she does not only know what kind of information the client is interested in, but she can also retrieve the answer the client receives. If the attacker is not eligible for the query of a client, she though may infer useful information from the volume of the encrypted answer returned to the client. We can prevent the attacker from reading the client's query by encrypting it. If the query itself is not encrypted using a hybrid scheme, nondeterministic encryption should be employed. To prevent inferences based on observed answer volume, the encrypted answer could be padded with dummy objects.

- An attacker can validate presumptions concerning source data she is not eligible to request. For this purpose, the attacker expresses her presumption in a query and misuses observed accreditation-certificates which are eligible in the query context. The attacker is unable to decrypt the answer, but merely based on the volume of the answer she is able to validate her presumption. Sources that want to avoid this attack avenue should accept queries only if the queries are signed, where the signatures are verifiable using the verification keys in the accompanying accreditation-certificates. To prevent inferences based on observed answer volume, the encrypted answer could be padded with dummy objects. Source repository reconstruction based on exhaustive guessing should be detectable.

- If any participant misused someone else's accreditation-certificates, the correct owner could be erroneously or maliciously blamed for the request. However, in a dispute about the sender of the request, no one can exhibit any essential evidence pro or contra the blame. If there is an interest in documenting the sender of a request, the protocol must be extended by appropriately signing the request.

  Attacks based on replaying signed requests, such as denial of service attacks, will still be accounted to the signing party, as long as no proof of the request's freshness is included.

- Further concerns about accountability or requirements on integrity could be dealt with by additional actions that are based on appropriate signing. These actions would be founded on the identity certificates offered by the basic informational environment to achieve accountability or integrity, respectively. The use of identity certificates impairs the signer's anonymity.

- In particular, if a source is concerned about a client redistributing received data without the source's approval, the source can fingerprint the delivered copies of the data, e.g., images or sounds.

- There are no specific provisions or additional obstacles to availability. Since authorization is based on accreditation-certificates, availability of the corresponding public verification keys is required.

6.3.4. The Design of Secure *I*-Mediation

We now extend the approach presented in Section 6.3.3 for the case of i-mediation. To begin with, we ignore security requirements for the moment and just state a rough abstract **protocol for mediated query answering**. The protocol can be divided into two phases:

Request phase

a) A client C sends a global query q to an i-mediator M.

b) The i-mediator M decomposes the query q into a set of subqueries q_S, where the subquery q_S is supposed to be appropriate for some source S.

c) The i-mediator M sends the subquery q_S to the source S, for each relevant source S.

Delivery phase

d) Each relevant source S evaluates its subquery q_S and produces a subanswer consisting of data d_S.

e) Each relevant source S sends its subanswer d_S back to the i-mediator M.

f) The i-mediator M integrates the received subanswers d_S into a global answer d.

g) The i-mediator M sends back the global answer d following the directions given by the return information.

Now taking care of the security properties achieved so far, we can easily combine secure direct querying with i-mediation. In the straightforward case the preparatory phase may remain unchanged. However, we will introduce a change to the preparatory phase when describing the architecture in Section 6.4. In the query phase, we build a new protocol for secure mediated query answering from the protocol for secure query answering and the protocol for mediated query answering. The latter is modified by applying the former on the communication between the client and the mediator as well as on the communication between the i-mediator and the sources, as illustrated in Figure 6.4:

- In step a) the client includes her accreditation-certificates into the request for query q.
- In step c), for each of the relevant sources, the i-mediator selects a subset of accreditation-certificates which is necessary to answer the subquery q_S. Then these "local accreditation-certificates" are attached to q_S and sent to source S. As a simple default, the i-mediator just forwards the received set of accreditation-certificates.
- In step d) each relevant source performs the security actions of step 2 of the protocol for secure query answering, and query evaluation is restricted as stated in step 3 of the protocol for secure query answering.
- Finally, in step e) each relevant source first encrypts its subanswer according to step 4 of the protocol for secure query answering before sending it back to the i-mediator.

It can be checked that the protocol for secure mediated query answering sustains the security properties achieved so far as for the simple case. There are only some minor restrictions. We note four aspects:

- Another participant, the i-mediator, acquires knowledge about the client's accreditation-certificates and thus could give raise to false blames about the sender of a request. The i-mediator could also redistribute accreditation-certificates of clients to third parties or misuse the client's accreditation-certificates to validate presumptions about the source's data. However, these possibilities do not introduce substantially new problems.
- An i-mediator may compromise the integrity of queries and answers, if no additional actions are taken. In particular, clients have to trust the i-mediator for the delivery of complete answers in the context of best effort i-mediation.
- Since i-mediators break a client's global query down to several subqueries for several sources, signatures protecting the entire global query will not be seen

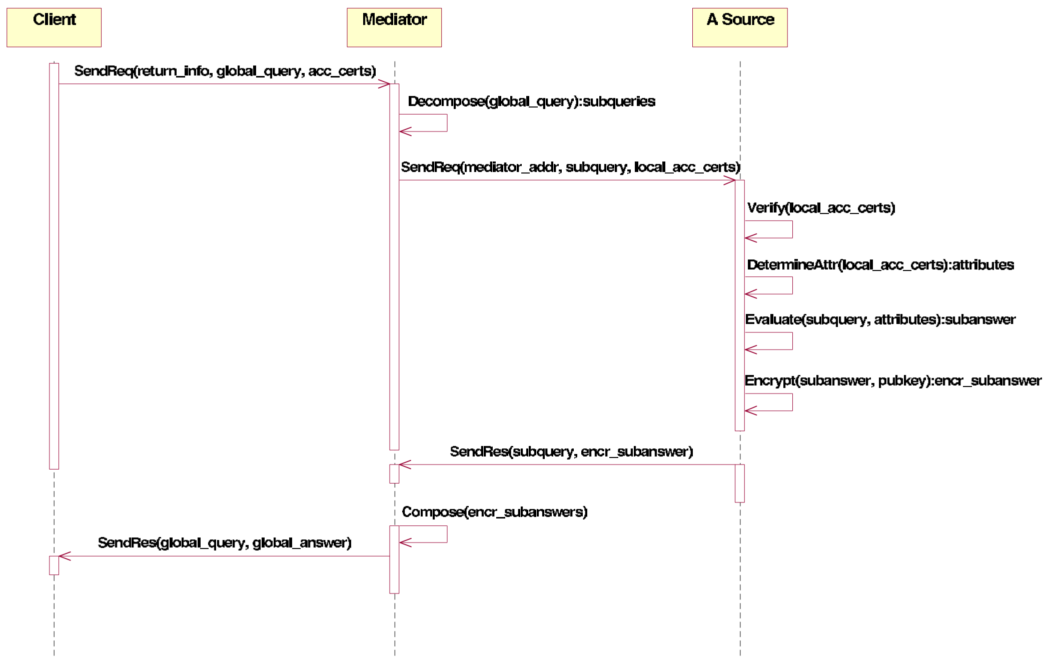

Client
Mediator
A Source
SendReq(return_info, global_query, acc_certs)
Decompose(global_query):subqueries
SendReq(mediator_addr, subquery, local_acc_certs)
Verify(local_acc_certs)
DetermineAttr(local_acc_certs):attributes
Evaluate(subquery, attributes):subanswer
Encrypt(subanswer, pubkey):encr_subanswer
SendRes(subquery, encr_subanswer)
Compose(encr_subanswers)
SendRes(global_query, global_answer)

Figure 6.4.: Protocol for secure mediated query answering

and verifiable by the sources. Also, the i-mediator is unable to break down a global query being encrypted as one unit into subanswers. Partial encryption and signing of the query is a topic for future research and is beyond the scope of this thesis.

- Since i-mediators construct a global answer for the client from several subanswers from several sources, each subanswer cannot be encrypted as one unit. Instead appropriate parts are encrypted individually. Supplying individually encrypted dummy parts as a padding in the answer to veil the answers' size is another topic for future investigation and is out of the scope of this thesis.

On the other hand, we can exploit the protocol for secure mediated query answering in favor of anonymity, since in general there is no need for a direct connection between the client and a source.

6.3.4.1. Layered *I*-Mediation

So far we treated the case that there is only one layer of i-mediation. However, we have argued that i-mediation does not essentially affect the security properties of direct querying, and thus we could use our approach also for i-mediation across several layers.

6.3.4.2. Integration of Subanswers – Functionality versus Confidentiality

During the delivery phase of the protocol for secure mediated query answering we are faced with the problem that the following requirements may be conflicting:

- The i-mediator has to integrate the subanswers sent back to the i-mediator by the sources.
- The i-mediator should not be able to break the security policies of the sources. In particular, the i-mediator should not gain protected information from the subanswers.

Our system fulfills the second requirement by encrypting the answers with the public keys of the client. Without a further thought this effectively disables fulfillment of the first requirement. Therefore, we assume that in general a source's security policy does not protect all of the source's data, i.e., parts of it are publicly available. In order to minimize the restrictions on functionality, we encrypt only the protected part of any subanswer. If a datum is protected, the i-mediator operates on the ciphertext only. If a datum is not protected, the i-mediator may operate on the unencrypted datum. In each case no trust in the i-mediator is necessary.

6.3.5. Summarized Security Properties

The protocols for secure query answering and for secure mediated query answering as proposed in Sections 6.3.3 and 6.3.4 provide different security properties during the request phase and during the delivery phase.

Basically, during the request phase the sources are provided with the certified information required to autonomously enforce authorization and to ensure the confidentiality of the answer. At the same time misused certified information cannot be turned against its owner regarding the content of the accompanying query.

Without the proposed extensions the basic request phase is open to some attacks that may be of concern, depending on the application environment. During the request phase the proposed extensions optionally provide query confidentiality, query integrity and query accountability.

For the basic delivery phase the protocols ensure the confidentiality of the answer, such that only the owner of the certified information accompanying the request can reveal the answer. During the delivery phase the proposed extensions provide for answer integrity, answer accountability and traceability for some kinds of answer objects, e.g., images or sounds.

During both phases the basic protocols do not require the use of information that may identify the client. Thus, the protocols are compatible with additional dedicated techniques to achieve client anonymity. If client anonymity is an important security property, the simple extensions proposed in Section 6.3.3 providing client accountability should not be used.

Use of accreditation-certificates avoiding the identities of the clients at first glance seems to support anonymity. Eventually, the public keys being included in an accreditation-certificate are also linkable facts about the client, even if they are not directly linkable to a certain client. A diligent privacy violator observing the movement of an accreditation-certificate by linking observations of its included public keys may still be able to build a dossier of an individual. Also i-mediation raises some additional concerns regarding the privacy of the clients. I-Mediation requires the clients to pass their requests through i-mediators. This means that the i-mediators can compile client profiles and misuse them. Once a client hands over her accreditation-certificate to the i-mediator, she cannot be sure the i-mediator retains it for no other purposes except for processing her query. Moreover, i-mediators will be amiable targets, for an i-mediator is a kind of channel where the accreditation-certificates of individuals conglomerate.

We observe that the privacy of the clients cannot be protected by using the accreditation-certificates if the client does not make proper use of public key technology. This requires the client's awareness of the implications when using accreditation-certificates. Using the same accreditation-certificate in many requests could be almost as privacy violating as using an identity certificate. Instead, a client can employ multiple accreditation-certificates specifying the same personal authorization attribute(s) in each request. Another solution is the employment of unlinkable and untraceable private-certificates (see Section 4.3.2) instead of using accreditation-certificates.

The protocols do not provide availability-related provisions. *I*-Mediation does not raise additional concerns with respect to availability of the involved service parties, i.e., the *i*-mediator, the sources and the trusted authorities.

All security properties of the protocols leverage off the authenticity and integrity of accreditation-certificates and identity-certificates, respectively. The clients and sources therefore need to place trust in the basic informational environment described in Section 6.3.2. The clients also have to trust the sources to provide answers to their best knowledge, and they need to trust the *i*-mediator to perform a best effort in providing complete answers. Data providers have to trust the sources to perform proper authorization as well as to ensure answer confidentiality and traceability. The sources have to trust the clients not to redistribute confidential answer objects. However, object redistribution by clients is traceable for some kinds of objects, e.g., images or sounds.

6.4. The Overall Architecture of Secure *I*-Mediation

The *Multimedia Mediator*, MMM, is implemented as an autonomously operating agent. It is based on the object-oriented database management system O2 [11]. Figure 6.5 shows its environment and its functional and security architecture. Seen from the client, the MMM appears as an object-oriented database which has 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 client addresses the MMM by means of her personal *user agent*. Data sources are connected to the MMM with the help of *wrapper agents*. Interoperability is supported by employing CORBA [67] for data exchanges, KQML [38] for agent communications, and ODL/OQL [24] for schema declarations and query expressions.

This architecture supports the design for secure *i*-mediation presented in Section 6.3 in a straightforward way as follows. A client can send a *request*, i.e., a query accompanied by accreditation-certificates, to the *i*-mediator via her user agent. The MMM checks whether the client is authorized to access the data necessary to answer the query. If the latter is the case, the MMM decomposes the query and sends *subrequests*, i.e., subqueries accompanied by an appropriate subset of the client's accreditation-certificates, to the sources via the wrapper agents. The sources may check whether the client is authorized to access the data necessary to answer the subquery. If the latter is the case, the sources send back their subanswers to the i-mediator, which in turn integrates the subanswers and sends the resulting global answer back to the client.

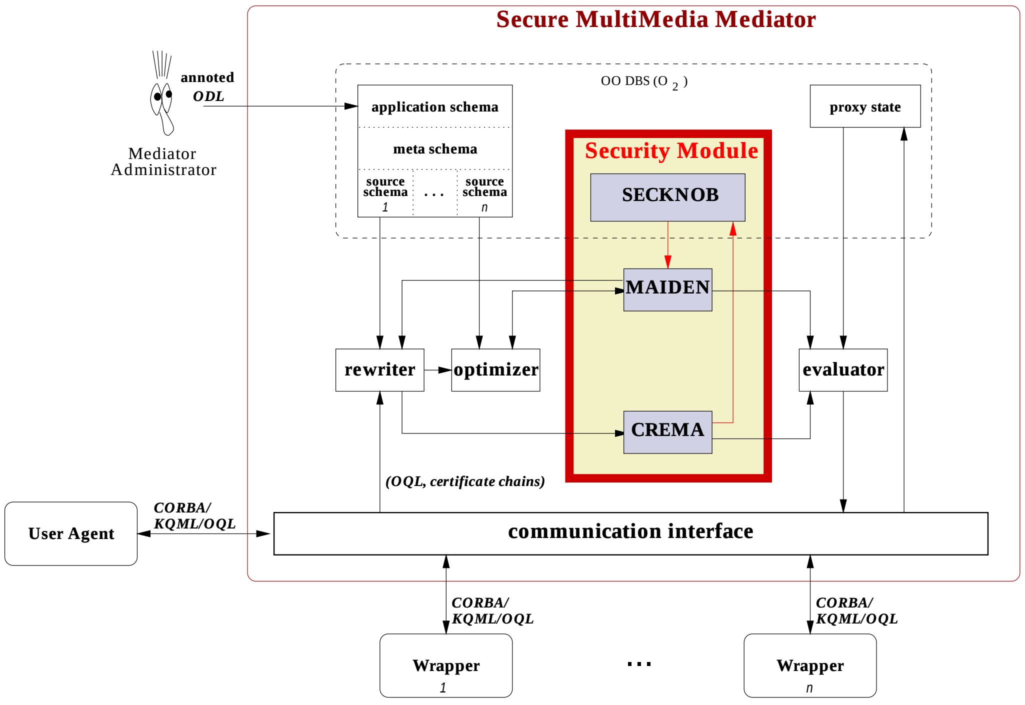


Figure 6.5.: Security architecture of the Multimedia Mediator

In order to implement the MMM-specific secure i-mediation two aspects need to primarily be elaborated:

- We need an *authorization model* that allows considering expressions over personal authorization attributes extracted from accreditation-certificates as grantees, to be used for representing the query access authorizations of the sources. As noted in Section 6.3.2, we assume that the participants of MMM-specific i-mediation agree on a common understanding of personal authorization attributes. In this case, the personal authorization attributes are implicitly used as special bound authorization attributes, i.e., in the query access authorizations. The only difference to the bound authorization attributes discussed in Section 4.4.1 is that these special bound authorization attributes are not delivered to other entities (e.g. grantees or delegatees).

- We need to specify the query access authorizations of the sources. Since we use ODL for schema declarations, the specifications of the query access authorizations should be pertinent to this application of ODL.

The elaboration of these aspects are handled in the following sections. In Section 9, we will provide more details on how secure *i*-mediation is actually implemented as part of the MMM.

6.4.1. Authorization Model

In Section 3.2, we have remarked that in *i*-mediation a client is characterized by her assigned properties, and each source follows a security policy expressed in terms of characterizing properties. In the case of MMM-specific secure *i*-mediation, a client is characterized by her personal authorization attributes, and the security policy of each data source is expressed in terms of these personal authorization attributes. Moreover, in Section 4.5.1, we have presented the property conversion policy (i.e. that maps the free properties on bound properties) as part of a grantor's whole security policy. It is important to note that in the context of MMM-specific secure *i*-mediation, the grantors do not need to specify such property conversion policies, since the personal authorization attributes are implicitly used as bound authorization attributes (see Section 4.4.1).

Accordingly, we present in the following an authorization model that explains what we mean by "a client is authorized to access some data" in the context of MMM-specific secure *i*-mediation. What we want to do precisely is to decide whether a client u may access a data object o by the method m, i.e., whether she is allowed to exercise the *privilege* $p = (o, m)$.

In this context, the client u is modelled by the conjunction r of the personal authorization attributes contained in the accreditation-certificates included in the client's request. The conjunction r corresponds to identified users in more traditional approaches. For deciding whether r represents a client who is allowed to exercise the privilege p, the i-mediator as well as the sources need two features:

- An *authorization database* of privileges granted to appropriately represented grantees, and
- *authorization policies* to evaluate the request with respect to the authorization database.

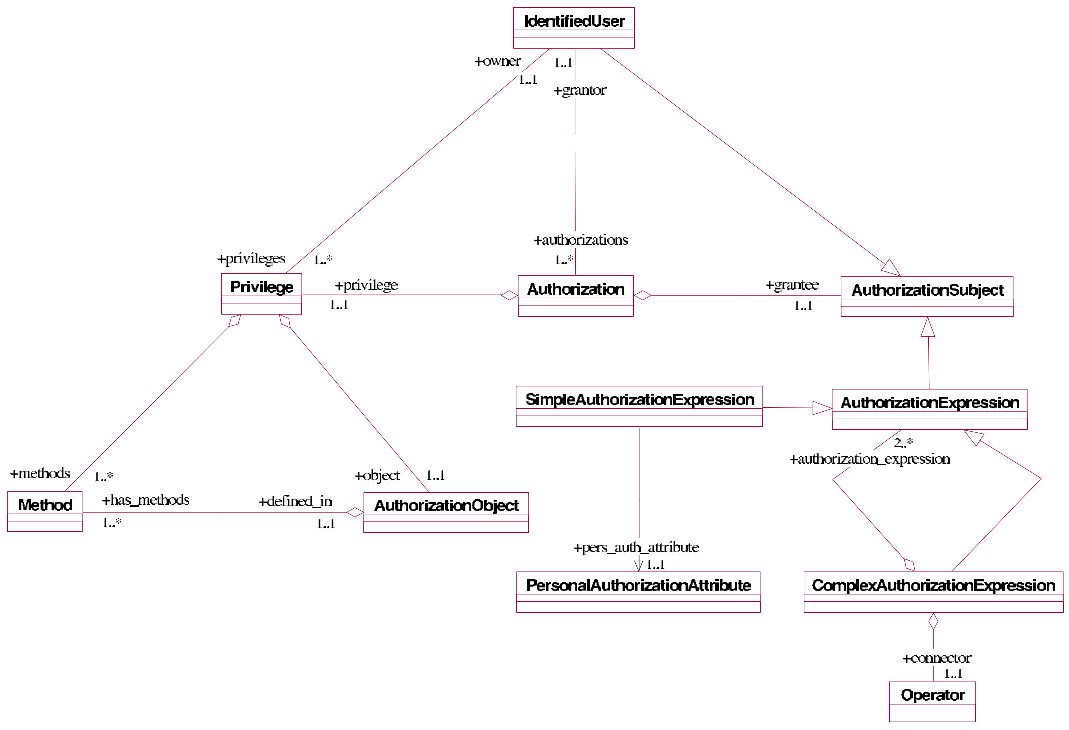


Figure 6.6.: A class model for the authorization database

Figure 6.6 shows a class model for the authorization database. As usual, an *authorization* specifies a grantor, a privilege and a grantee:

- A privilege is composed of an 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 sources, respectively).

- The *grantee* might be an identified user, as usual, or alternatively, as a particularity of our property-based authorization model, a *Boolean expression over personal authorization attributes,* or *true*, or *false* (*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.* An authorization expression over personal authorization attributes might be a *simple authorization expression* or a *complex authorization expression.* The former is associated with only one personal authorization attribute. The latter consists of at least two authorization expressions, each of which might be a simple or a complex authorization expression, and are connected by one of the logical operators AND or OR. Note that a client's representation r, being a conjunction of the personal authorization attributes built from the client's accreditation-certificates, is an authorization expression, too.

A particular instance (p, g, a) of an authorization with privilege p, grantor g, and authorization expression a as grantee means that

g has granted privilege p to the class of clients
who can represent themselves
by some authorization expression r
such that "r *qualifies for* a".

The relationship *"qualifies for"* formalizes the rough intuition of *"logically implies* under additional assumptions". In the case of MMM-specific secure i-mediation, the only "additional assumption" is that the MMM-specific i-mediation participants share a common vocabulary for the personal authorization attributes. The specification of the authorization expression a as *true* means that "any r *qualifies for* a", while *false* states that "no r *qualifies for* a". The nonexistence of a grantee is interpreted as *false.*

Expressed in terms of role-based access control, each conjunction of the authorization expression a corresponds to a *role*, assignment of which requires this conjunction to be qualified by the authorization expression r which constitutes also a conjunction.

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 checks the involved qualifications only, 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 *i*-mediator itself applies two more elaborated *authorization policies for i-mediation*, one for querying schemas and another one for querying instances, which combine the involved qualifications with additional considerations, to be presented in Section 6.4.3.

6.4.2. 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 convertible into a canonical form expressed in ODL. For the current prototype, we use the following basic ODL features: *class*; *class attribute*[2]; *attribute type* built from classes and atomic types (*boolean, string, long*, . . .) using constructors such as *set, list, struct*; *relationship* (specifying inverse attributes); *key*; *extent* for denoting an access path to a full class extension. Further advanced features (e.g. *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 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.
2. We assume some generic access method for all authorization objects (refraining from specializing the generic method into *read, navigate*, and so on).

[2]The ODL term is *attribute*. We use the term *class attribute* instead in order to distinguish it from the term *personal authorization attribute*.

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.

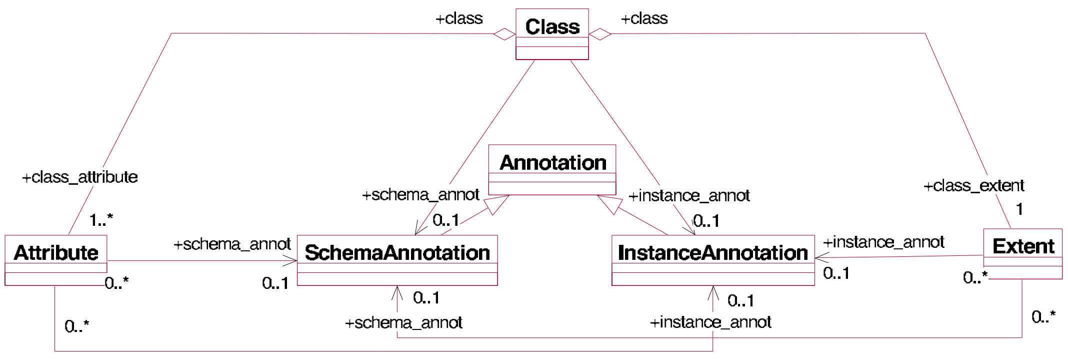


Figure 6.7.: A class model showing the relationships between annotations and ODL features

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 (see Figure 6.7):

- 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 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 feature.
- 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 an element *ele* of a struct-declaration of an attribute type grants the privilege to access that element of an instance object provided the struct part has been accessed before.
- An *instance access annotation* on an extent *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 the following section we define the authorization policies which finally regulate the actual access decisions.

6.4.3. MMM-Specific Secure *I*-Mediation

For secure *i*-mediation as implemented for the MMM, we distinguish the initialization phase and, afterwards, preparatory phases for sources and clients, respectively, and request phases and corresponding delivery phases. In the following the basic concepts of these phases are outlined. Figure 6.8 visualizes some of these concepts.

Throughout this section we use a running example based on the motivating example scenario exposing the psychoanalyst's query outlined in Section 6.2. This example will help to understand the tasks handled in each phase.

6.4.3.1. Initialization Phase

In the *initialization* phase, an *i*-mediation administrator *instantiates* the MMM by declaring an *application schema* expressed in ODL which captures the information needs of the anticipated clients. For many parts of the application schema, there

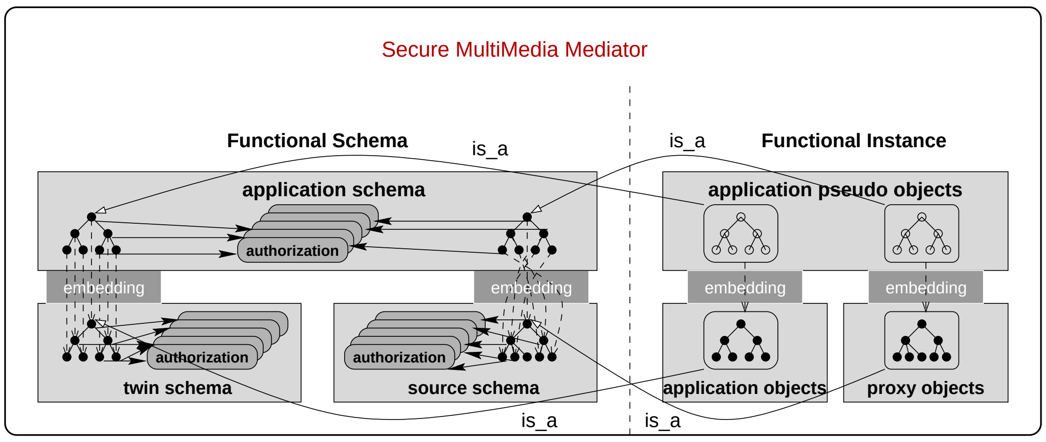


Figure 6.8.: Schema and instance with authorizations

will be no persistent instance in general, but the corresponding data is dynamically retrieved from sources at query time.

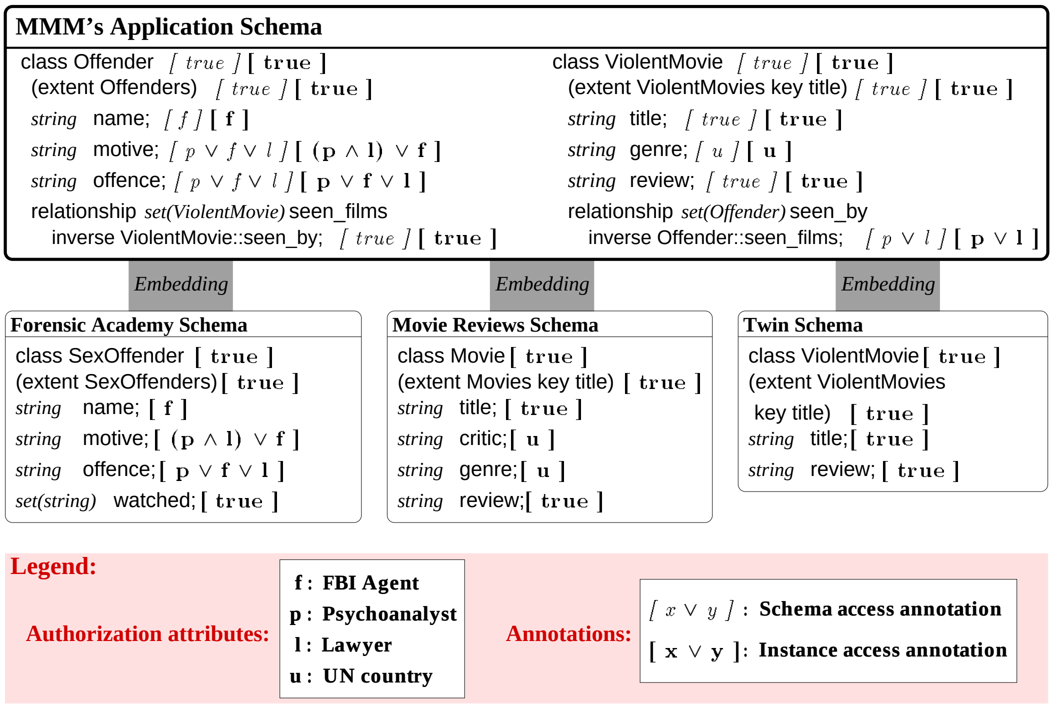
But for some parts of the schema, the administrator may want to maintain data owned by the MMM itself. Technically, for such parts the administrator additionally declares a *twin schema* as a subschema of the application schema, and she inserts an instance for this twin schema. Later on, query processing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

While specifying the application schema and the twin schema, the administrator also grants the *authorizations* using annotated ODL declarations as introduced in Section 6.4.2 for both schemas. Conceptually, the authorizations for both the application schema and the twin schema are stated independently, though, in practice, they usually will be closely related.

6.4.3.2. 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 complies with the functional requirements of the instantiated MMM. Furthermore, the authorization policy of the source must be based on accreditation-certificates. 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 i-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 [8]. 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. For the sake of simple presentation, we only present the instance access annotations of the source schemas.



MMM's Application Schema

```
class Offender  [ true ] [ true ]
  (extent Offenders)  [ true ] [ true ]
  string  name;  [ f ] [ f ]
  string  motive;  [ p ∨ f ∨ l ] [ (p ∧ l) ∨ f ]
  string  offence;  [ p ∨ f ∨ l ] [ p ∨ f ∨ l ]
  relationship set(ViolentMovie) seen_films
     inverse ViolentMovie::seen_by;  [ true ] [ true ]

class ViolentMovie  [ true ] [ true ]
  (extent ViolentMovies key title)  [ true ] [ true ]
  string  title;  [ true ] [ true ]
  string  genre;  [ u ] [ u ]
  string  review;  [ true ] [ true ]
  relationship set(Offender) seen_by
     inverse Offender::seen_films;  [ p ∨ l ] [ p ∨ l ]
```

Forensic Academy Schema

```
class SexOffender  [ true ]
(extent SexOffenders) [ true ]
string   name;  [ f ]
string   motive; [ (p ∧ l) ∨ f ]
string   offence; [ p ∨ f ∨ l ]
set(string)   watched; [ true ]
```

Movie Reviews Schema

```
class Movie [ true ]
(extent Movies key title)  [ true ]
string   title;  [ true ]
string   critic; [ u ]
string   genre; [ u ]
string   review; [ true ]
```

Twin Schema

```
class ViolentMovie [ true ]
(extent ViolentMovies
 key title)   [ true ]
string   title; [ true ]
string   review; [ true ]
```

Legend:

Authorization attributes:
f : FBI Agent
p : Psychoanalyst
l : Lawyer
u : UN country

Annotations:
[$x \vee y$] : **Schema access annotation**
[**x ∨ y**] : **Instance access annotation**

Figure 6.9.: Example for the initialization phase for the MMM and preparatory phases for the sources

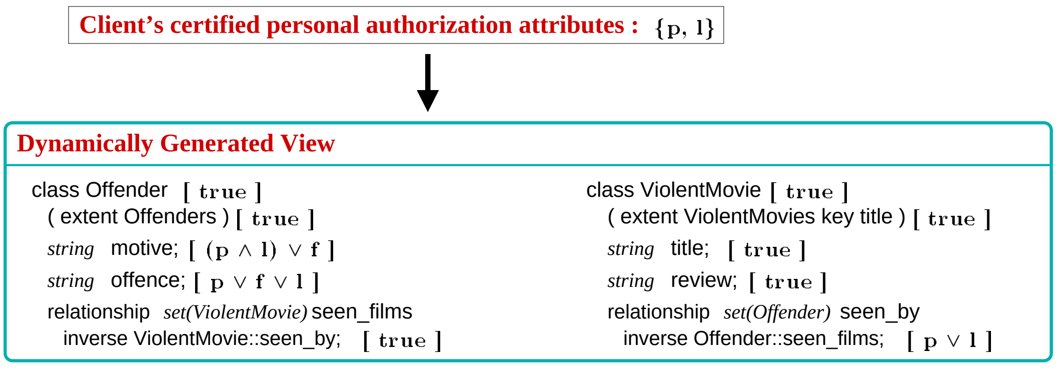
Figure 6.9 illustrates an annotated application schema, two external annotated source schemas as well as an annotated twin schema. In this example, the application schema of the instantiated i-mediator consists of two classes: `class Offender` and `class ViolentMovie`. The application schema is embedded into two external source schemas, a forensic academy schema and a movie reviews schema, and also a twin schema which is considered as a subschema of the application schema. This example only considers schema access annotations and instance access annotations over the certified personal authorization attributes *FBI agent, psychoanalyst, lawyer* and *UN country*.

As it is shown in Figure 6.9, the administrator of the *i*-mediator has declared different schema access annotations and instance access annotations to the specific parts of the application schema and twin schema. According to these annotations, the declaration and the corresponding instance attribute values of `Offender::name` can only be revealed to *FBI agents*. While the declaration as well as the instance attribute values of `Offender::offence` and the declaration of `Offender::motive` can be revealed to any client showing at least one of the personal authorization attributes *psychoanalyst*, *FBI agent*, or *lawyer*, the instance attribute values of `Offender::motive` can be revealed to any client being at least a *psychoanalyst* and a *lawyer*, or an *FBI agent*. The declaration as well as the instance attribute values of `ViolentMovie::genre` can only be disclosed to the citizens of the UN countries. The extensions `Offenders` and `ViolentMovies` can be seen by any client.

6.4.3.3. Preparatory Phase for a Client

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

- A subschema is *permitted* (for an authorization expression r) 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 [68, 26]).


Client's certified personal authorization attributes : {p, l}
Dynamically Generated View

```
class Offender [ true ]
  ( extent Offenders ) [ true ]
  string  motive; [ (p ∧ l) ∨ f ]
  string  offence; [ p ∨ f ∨ l ]
  relationship  set(ViolentMovie) seen_films
    inverse ViolentMovie::seen_by;   [ true ]

class ViolentMovie [ true ]
  ( extent ViolentMovies key title ) [ true ]
  string  title;  [ true ]
  string  review;  [ true ]
  relationship  set(Offender) seen_by
    inverse Offender::seen_films;   [ p ∨ l ]
```

Figure 6.10.: Example for the preparatory phase for a client

Thus, the client gets a dynamically generated *external view* about the *functionality* of the *i*-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 *i*-mediation. Hence, we favour this novel dynamic approach rather than declaring a limited set of external views in advance.

On demand, the client also gets the *authorization requirements* for the corresponding (virtual) instances, i.e., the client 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 client can collect accreditation-certificates with her personal authorization attributes *potentially* providing evidence of her eligibility to see answers to submitted queries.

The largest permitted subschema is treated similarly to the subanswers produced in the delivery phase and encrypted according to the *answer encryption policy for i-mediation* (see Section 6.4.3.5). In order to do so, we have to interprete the schema access annotations as fictitious instance access annotations on the meta schema. Thus, the client gets a protected subschema.

Figure 6.10 illustrates the example view dynamically generated from the application schema. Since the client is a *psychoanalyst* and a *lawyer*, and does not possess certified *FBI agent* and *UN country* attributes, she is allowed to see a view hiding the existence of the class attributes `Offender::name` and `ViolentMovie::genre`.

6.4.3.4. Request Phase

In a *request phase*, a prepared client sends a global *request* to the MMM. A global request includes a global query with respect to the application schema and a set of accreditation-certificates. 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 subquery with respect to the internal representation of the source schema and appropriately selected accreditation-certificates, 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 i-mediation*:

- A request is (globally) *permitted* (for an authorization expression r) 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 accreditation-certificates. 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 to and autonomously evaluated by the sources. Otherwise the request is (globally) *denied*, and the client will be informed that no answer can be returned due to the lack of certain 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 *i*-mediator and forwarded to the client without any further restrictions.

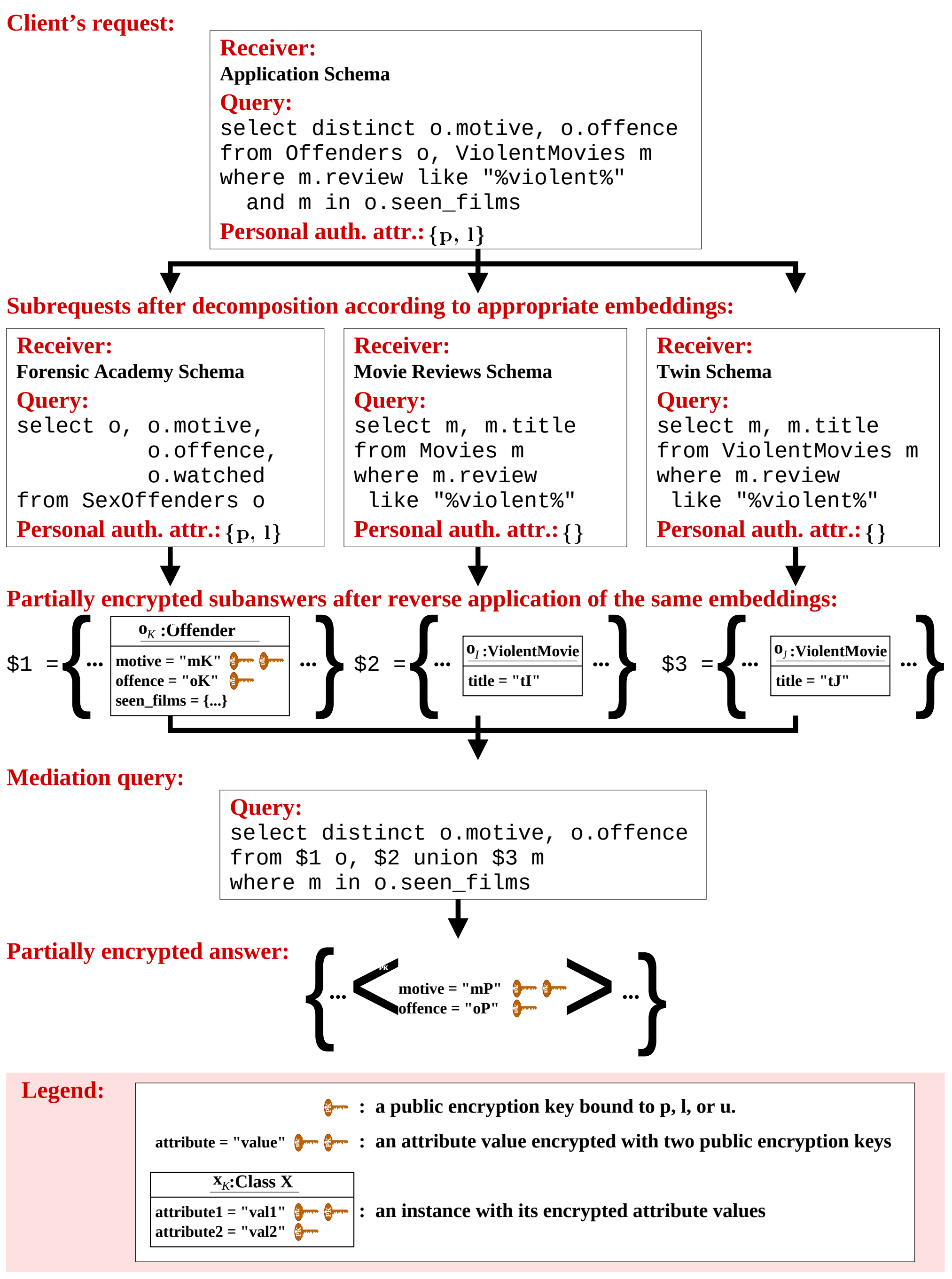
Thus, seen from the client, the authorization requirements of the *i*-mediator and the sources are *conjunctively* combined: access to source data via the *i*-mediator is permitted iff it is allowed by *both* the *i*-mediator *and* the source. Accordingly, the security module of the *i*-mediator acts 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.

Figure 6.11 shows a formalization of the psychoanalyst's request including a query and her certified personal authorization attributes. A possible decomposition of the query consists of three subqueries, each of which is part of a subrequest to the respective source, together with the personal authorization attributes necessary to answer the subquery.

6.4.3.5. Delivery Phase

Each request phase is followed by a corresponding *delivery phase* (allowing the interleaving of several requests). In this phase, a source autonomously applies its own authorization policy on the subrequest and thereby produces a *protected subanswer*. As a default, we assume that the source applies the *pure authorization policy*, as introduced in Section 6.4.1, and encrypts the subanswer accordingly. More specifically, the source is supposed to proceed according to the following *answer encryption policy for i-mediation*:

- Within the answer each occurence of a value v of an atomic type, appearing under a class attribute *attr* (as direct attribute value or as an element of it), is *nondeterministically encrypted* with a subset of the encryption keys that are contained in the submitted accreditation-certificates.
- The pertinent subset of encryption keys is determined on the basis of a syntactic analysis from which parts of the stored instance an *information flow* into



Client’s request:

Receiver:
Application Schema
Query:
```
select distinct o.motive, o.offence
from Offenders o, ViolentMovies m
where m.review like "%violent%"
  and m in o.seen_films
```
Personal auth. attr.: {p, l}

Subrequests after decomposition according to appropriate embeddings:

Receiver:
Forensic Academy Schema
Query:
```
select o, o.motive,
          o.offence,
          o.watched
from SexOffenders o
```
Personal auth. attr.: {p, l}

Receiver:
Movie Reviews Schema
Query:
```
select m, m.title
from Movies m
where m.review
 like "%violent%"
```
Personal auth. attr.: { }

Receiver:
Twin Schema
Query:
```
select m, m.title
from ViolentMovies m
where m.review
 like "%violent%"
```
Personal auth. attr.: { }

Partially encrypted subanswers after reverse application of the same embeddings:

$1 = { ... o_K :Offender motive = "mK" offence = "oK" seen_films = {...} ... }
$2 = { ... o_I :ViolentMovie title = "tI" ... }
$3 = { ... o_J :ViolentMovie title = "tJ" ... }

Mediation query:

Query:
```
select distinct o.motive, o.offence
from $1 o, $2 union $3 m
where m in o.seen_films
```

Partially encrypted answer:

Legend:

: a public encryption key bound to p, l, or u.
attribute = "value" : an attribute value encrypted with two public encryption keys
x_K:Class X attribute1 = "val1" attribute2 = "val2" : an instance with its encrypted attribute values

Figure 6.11.: Example for request and delivery phases

the occurence of v might have happened. This analysis is performed on the granularity and in terms of the schema (or any other applicable data description). Roughly speaking, expressed in the selected subset of ODL, these parts comprise the classes, class attributes and relationships, elements of struct-declarations and extents, the instances of which are needed to be accessed in order to select and to retrieve the occurence of v. Each of these schema items carries an authorization expression as an instance access annotation, the conjunction of which is here denoted by a. The actual access to all these instance items has been permitted since the client has represented herself by a conjunctive authorization expression r qualifying for a. Now, the personal authorization attributes in r, or any subset of them still qualifying for a, show the client's eligibility to see the information (potentially) contained in the occurence of the value v within the answer. Thus, the encryption is done with the subset of encryption keys which are bound to these authorization attributes by the submitted accreditation-certificates.

- All *structural parts* of the answer 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. Afterwards, the i-mediator produces a global protected answer. In the straightforward case, the i-mediator again proceeds according to the answer encryption policy for i-mediation, using the instance access annotations of the application schema. Of course, the i-mediator does not have to encrypt an occurence of a value repeatedly with the same key.

In our example, shown in Figure 6.11, the three subrequests result in two sets of application pseudo objects representing movies, the attributes named `title` of which are unencrypted, and a set of `Offender` objects with their `motive` and `offence` attribute values being encrypted. The elements of their `seen_films` sets are `Violent-Movie` object references. The titles of the referenced `ViolentMovie` objects are the

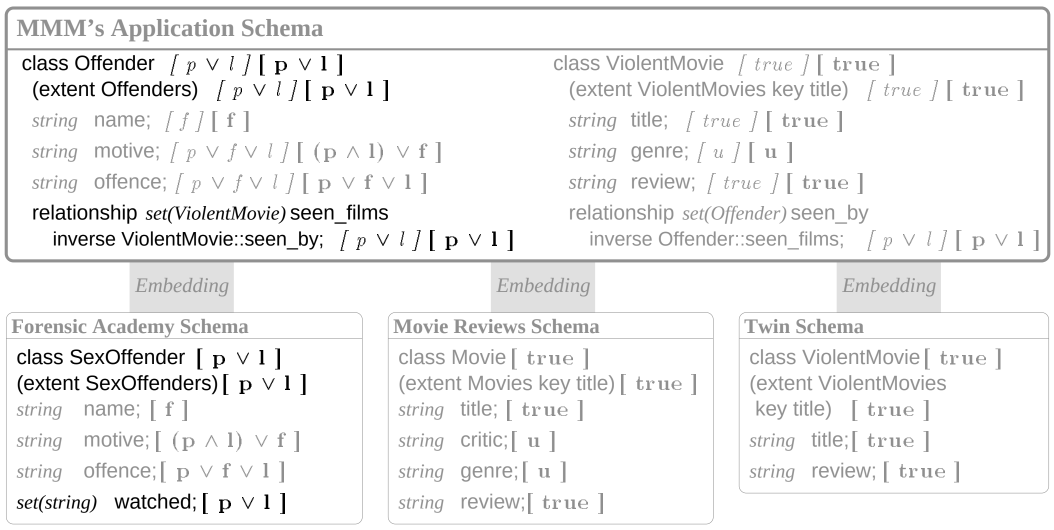

MMM's Application Schema
class Offender [$p \vee l$] [$\mathbf{p} \vee \mathbf{l}$]
(extent Offenders) [$p \vee l$] [$\mathbf{p} \vee \mathbf{l}$]
string name; [f] [$\mathbf{f}$]
string motive; [$p \vee f \vee l$] [$(\mathbf{p} \wedge \mathbf{l}) \vee \mathbf{f}$]
string offence; [$p \vee f \vee l$] [$\mathbf{p} \vee \mathbf{f} \vee \mathbf{l}$]
relationship set(ViolentMovie) seen_films
inverse ViolentMovie::seen_by; [$p \vee l$] [$\mathbf{p} \vee \mathbf{l}$]
class ViolentMovie [true] [true]
(extent ViolentMovies key title) [true] [true]
string title; [true] [true]
string genre; [u] [$\mathbf{u}$]
string review; [true] [true]
relationship set(Offender) seen_by
inverse Offender::seen_films; [$p \vee l$] [$\mathbf{p} \vee \mathbf{l}$]
Embedding
Embedding
Embedding
Forensic Academy Schema
class SexOffender [$\mathbf{p} \vee \mathbf{l}$]
(extent SexOffenders) [$\mathbf{p} \vee \mathbf{l}$]
string name; [$\mathbf{f}$]
string motive; [$(\mathbf{p} \wedge \mathbf{l}) \vee \mathbf{f}$]
string offence; [$\mathbf{p} \vee \mathbf{f} \vee \mathbf{l}$]
set(string) watched; [$\mathbf{p} \vee \mathbf{l}$]
Movie Reviews Schema
class Movie [true]
(extent Movies key title) [true]
string title; [true]
string critic; [$\mathbf{u}$]
string genre; [$\mathbf{u}$]
string review; [true]
Twin Schema
class ViolentMovie [true]
(extent ViolentMovies key title) [true]
string title; [true]
string review; [true]

Figure 6.12.: An alternative scenario with more restricted security policies

unencrypted elements of the returned `SexOffender::watched` sets. Now the set of `Offender` objects is joined with the union of the two `ViolentMovie` object sets in a i-mediation query, which delivers pairs of the encrypted `motive` and `offence` values as the global result.

Surely, the functionality of this phase is essentially restricted by the attribute values being encrypted with public keys of the client. 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. There are several strategies to avoid the restrictions:

- We try to push selections into the subqueries, and to transform the original query accordingly.
- We expect that usually some source items are not protected at all, i.e., there are no nontrivial authorization requirements. Formally, we annotate such items with *true* (interpreted as "*always allowed*"). Then any authorization expression qualifies for these annotations, and sources return the corresponding content data as plain text.
- We process encrypted data.

- We assign appropriate trust to the i-mediator.
- We shift the final integration of subanswers to the client who can decrypt the subanswers.

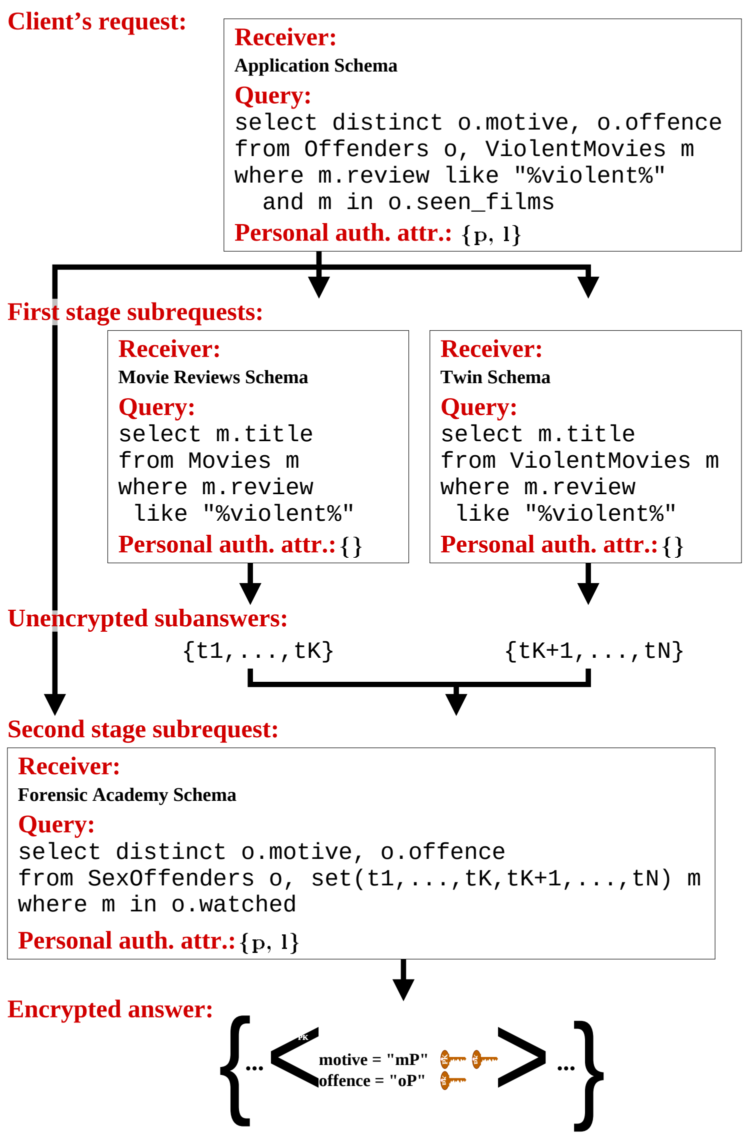
These topics require a separate investigation and are beyond the scope of this thesis, but we want to roughly illustrate the first strategy. See the modification of our example shown in Figure 6.12. The class `SexOffender` and its attribute `watched` are annotated by the authorization expression `p` ∨ `l` (and so are the corresponding elements in the application schema), in contrast to the original example. In consequence, the elements of the returned `SexOffender::watched` sets would be encrypted. Thus, the `ViolentMovie` objects referenced in the `Offender::seen_films` sets cannot be correctly related to the `ViolentMovie` objects resulting from the respective subrequests. Figure 6.13 depicts a possible solution for this problem. The result of the first stage subrequests is pushed to the forensic academy source for the second stage subrequest, which immediately produces the result of the global query.

6.5. Contributions

As the first contribution of this chapter, we discuss the security requirements for i-mediation with an emphasis on confidentiality and authenticity. We argue for basing the enforcement of these properties on certified personal authorization attributes rather than on identification. As the second contribution, we propose a general design of secure i-mediation where accreditation-certificates are roughly used as follows: clients show their eligibility for receiving requested information by the contained personal authorization attributes, and sources and the i-mediator guarantee confidentiality by using the contained encryption keys. As the third contribution of this chapter, we refine the general design for a specific approach to i-mediation and present the security architecture for the prototype of our Multimedia Mediator.

More concretely, we have the following contributions in this chapter:

- We discussed the needs of secure i-mediation, thereby emphasizing the differences and the new challenges in comparison to the more traditional federated approach.



Client's request:

Receiver: Application Schema

Query:

```
select distinct o.motive, o.offence
from Offenders o, ViolentMovies m
where m.review like "%violent%"
  and m in o.seen_films
```

Personal auth. attr.: {p, l}

First stage subrequests:

Receiver: Movie Reviews Schema

Query:

```
select m.title
from Movies m
where m.review
 like "%violent%"
```

Personal auth. attr.: {}

Receiver: Twin Schema

Query:

```
select m.title
from ViolentMovies m
where m.review
 like "%violent%"
```

Personal auth. attr.: {}

Unencrypted subanswers:

```
{t1,...,tK}          {tK+1,...,tN}
```

Second stage subrequest:

Receiver: Forensic Academy Schema

Query:

```
select distinct o.motive, o.offence
from SexOffenders o, set(t1,...,tK,tK+1,...,tN) m
where m in o.watched
```

Personal auth. attr.: {p, l}

Encrypted answer:

Figure 6.13.: Query evaluation for the alternative scenario

- We presented the protocols for *secure direct querying* and *mediated query answering*, and we examined their security properties.
- We analyzed the achievable security properties including support for anonymity, and we discussed the inevitable tradeoffs between security and *i*-mediation functionality.
- We presented the security architecture for the prototype of the MMM.
- We defined an *authorization model* that allows considering expressions over *personal authorization attributes* extracted from *accreditation-certificates* as grantees, to be used for representing the query access authorizations of the sources.
- We defined the specification of query access authorizations as *annotated schema declarations* within the framework of ODL.
- We presented a *schema authorization policy for mediation*, which allows dynamically generating external views for spontaneous users.
- We presented an *instance authorization policy for mediation*, which conjunctively combines the access restrictions of the mediator and the sources.
- We presented an *answer encryption policy for mediation*, which supports information flow control.
- We achieved an *isolated co-existence* of the functional layers and the security layers treating authorization data in close correspondence to functional data.

Part V.

Prototypical Demonstrators

7. The Agent PKI Framework

In this chapter we present the main features and components of an *agent PKI framework* which we have implemented to provide basic PKI services, such as issuing and verification of digitally signed certificates and credentials (see Sections 4.3.2 and 4.4.2). The agent PKI framework is a collection of tools which can be used to experiment with the basic PKI models and the hybrid PKI model (see Chapter 4).

When we started the implementation of the agent PKI framework, we had three design objectives in mind:

- The first objective was to provide appropriate PKI services for testing the basic functionality of the demonstrators presented in Chapters 8 and 9.
- The second objective was to support any arbitrary PKI-enabled application that requires PKI services. For this purpose, we provided appropriate application programming interfaces.
- The third design objective was to hide the complexity of the underlying security mechanisms, while facilitating service requests through simple service calls.

The details of the implementation are reported in two master's thesis [50, 58]. The implementation is partially based on investigations which resulted in a credential model (i.e. unifying the existing certificate and credential models [4, 51, 16, 31, 35]) which can be found in the master's thesis [57]. The class model representing the two faces – hidden (real) world and visible virtual views – of a property assignment process (see Figure 4.1 in Section 4.2.1) constitutes a reviewed core of the credential model which is reported in [57].

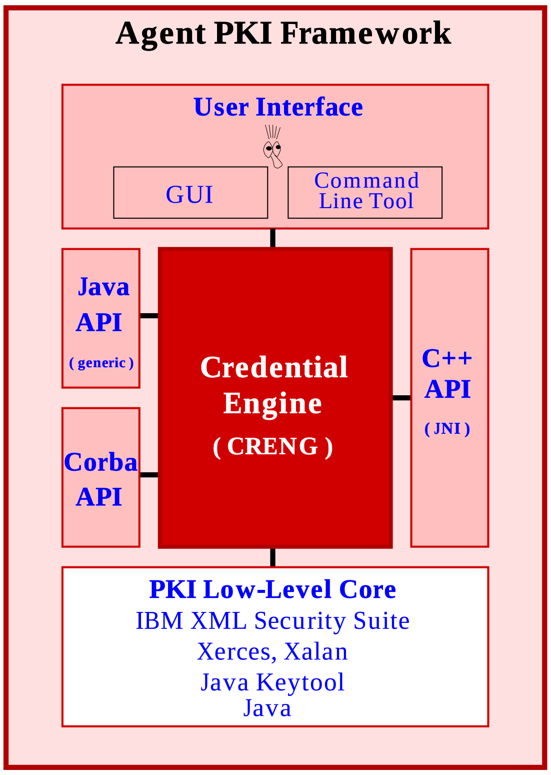


Figure 7.1.: High-level architecture of the agent PKI framework

7.1. The Full Architecture and The Main Features

The credential engine, CRENG for short, is the core module of the agent PKI framework. CRENG is implemented in Java on a Solaris platform. The PKI framework is universal, since it provides an application programming interface (i.e. implemented by using Java Native Interface) for arbitrary C++ applications and agents (e.g. our MultiMediaMediator agent). The framework also provides a CORBA interface for CORBA-enabled applications. Our implementation of the agent PKI framework is broad enough to allow the conduction of experiments with different PKI approaches and a hybrid PKI model, as described in Chapter 4. We proved the applicability of the agent PKI framework by employing it for the credential managers of the security modules presented in Chapters 8 and 9. Figure 7.1 visualizes the high-level architecture of the agent PKI framework.

In the following we outline the main features of the implemented PKI framework:

- Generating public/private key pairs, keystore and self-signed X.509 certificates. We use the X.509 certificates created by Java *keytool* [84], to obtain the public key stored in the digital certificate which is physically stored within the keystore.
- XML-encoding of certificates and credentials.
- Issuing of XML-encoded certificates and credentials.
- Verifying the signatures of certificates and credentials.
- Validity checking for certificates and credentials.
- Building chains of certificates and credentials, including licence-certificates and delegation-credentials, respectively.
- Evaluating chains of certificates and credentials (except of dealing with revocations so far).

The agent PKI framework also provides content management operations for certificates and credentials, including extracting the free properties and the bound properties. A GUI and command-line interface provided by this PKI framework facilitate the administration of certificates and credentials.

7.2. XML Encoding of Certificates and Credentials

We opted for an approach to certificates and credentials based on digitally signed XML documents [29, 74, 69, 5] in order to have freedom of experimenting and researching, and not to be constrained by any of the existing certificate and credential formats. Additionally, XML has the advantages of presenting self-describing documents and being widely used by various scientific disciplines. However, our approach is compatible with both X.509 and SPKI/SDSI philosophies. We have used IBM's *XML Security Suite* [5] for signing and verifying XML documents. XML Security Suite supports both DSA and RSA for digital signatures. Our implementation uses DSA for signing and SHA-1 for hashing. The XML Security Suite employs the XML parser *Xerces* and XSLT[1] stylesheet processor *Xalan* for the manipulation of XML documents [2].

We decided to design a common format for representing certificates and credentials. By using this format, it is possible to represent the certificates (except attribute-certificates) and credentials presented in Sections 4.3.2 and 4.4.2. In order to represent the attribute-certificates, the *subject* field must be extended to include a reference to an identity-certificate. Moreover, for the sake of simplicity, the *type* field indicates only the kind of the underlying document. This field could also be extended to encode the cryptographic algorithm used to generate the digital signature. Apart from these differences, the structure of a signed XML document is heavily based on the format presented in Section 4.2.1 and compires the following fields:

- a *subject* field which contains two public keys used for verification and encryption, respectively.
- a *content* field which textually describes the assigned properties. A property has a property name and a corresponding value, e.g., [`name`=FBIAgent, `value`=TRUE] or [`name`=age, `value`$\geq$18], etc.
- a field for a *responsible agent*: this field contains the public verification key of the agent which visibly represents the entity that is responsible for the property assignment and that has generated and digitally signed the XML document.
- a *type* field which indicates whether the signed XML document is a certificate or a credential.
- a *delegation* field[2] which is used only for credentials to indicate whether the credential under consideration is a bound property-credential or a delegation-credential.
- a *validity* field which encodes a certain time period during which the property assignment is valid.

[1]XSLT is a language for transforming XML documents into other XML documents.

[2]In the current implementation of the agent PKI framework, the *delegation* field of an XML document has the same semantic as the delegation bit used in the SPKI/SDSI approach in which a delegatee might use the encoded properties of a document for his own service requests.

- a *signature* field which contains a digital signature generated by using DSA and SHA-1.

As noted in Section 4.2.4, the "main document" and its "supporting documents" could form a *directed acyclic graph (dag)* with respect to their relationships concerning *support*. In order to focus on the major PKI services and to keep the PKI framework manageable, we employed *chains* instead of dags. A chain is also represented as an unsigned XML document which contains a set of signed XML documents. For the verification of certificate chains and credential chains, we have implemented algorithms which we have adopted from [4, 51] and [31, 35], respectively.

7.3. Contributions

In this chapter we presented the architecture and the main features of an *agent PKI framework* which provides basic PKI services such as encoding, issuing, and verifying certificates and credentials. The PKI framework supports a variety of PKI-enabled applications by providing a Java API and a C++ API. The framework also provides a CORBA interface for CORBA-enabled applications. We also presented the structure of digitally signed XML documents which are used to represent certificates and credentials.

8. An Agent-Oriented Hybrid PKI Demonstrator

In Chapter 4, we have presented a hybrid model to a PKI. In order to implement the general design of the hybrid model, a specific approach to the hybrid model has to be taken into account. For this purpose, we implemented an agent-oriented and KQML-based hybrid PKI demonstrator [50].

With the implementation of the demonstrator, we intended to prove the key ideas of the hybrid PKI model. In order to have freedom of experimenting and researching, and not to be constrained by the development stage of the existing *i*-mediator agent MMM, we have decided to develop a separate agent-based infrastructure (i.e. which constitutes a testbed) for demonstrating the basic functionality of the hybrid PKI model. We established such a testbed using our existing approaches and the corresponding tools:

- We followed and extended the basic approach of the security module (see Section 9) and the communication interface of the MMM agent.

- We bound the agent PKI framework (see Section 7) to the agent-based infrastructure in order to be able to experiment with digitally signed certificates and credentials and to implement a challenge-response algorithm utilizing digital signatures.

By this means, we achieved a scalable *agent core* which can be easily instantiated for different kinds of agents involved in an instance of the hybrid model. In the following sections we first present a hybrid mediation scenario underlying the demonstrator and then the architectural and implementational features of the demonstrator.

8.1. The Underlying Scenario

For the design of specific i-mediation, we have assumed that the i-mediation participants agree on a common understanding of "personal authorization attributes" (see Section 6). Based on this assumption, we could concentrate on the other design and implementational issues for secure i-mediation with an emphasis on confidentiality and authenticity.

For the hybrid PKI demonstrator we simulate a *hybrid mediation* scenario. An outline of such a scenario was given in Section 4.1. The need for utilizing i-mediation and f-mediation in a hybrid manner arises particularly from a conceptual challenge concerning how remote and autonomous i-mediation participants can agree on a common understanding of "personal authorization attributes". On the one hand, trusted authorities assign personal authorization attributes as *free properties*, in principle without knowing their later usage. And on the other hand, data sources and i-mediators independently define their security policies using bound authorization attributes (see Section 4.4.2) as *bound properties*. Surely, a client obtaining a free property assigned would prefer to look on the corresponding accreditation-certificate as if it was a bound property-credential expressing a permission to get services of a specific kind. But, unfortunately, in general a trusted authority is not at all related with the owners of sources and i-mediators. Also clients wish to be assisted by an i-mediator to assemble appropriate bound properties within suitable credentials in order to receive the information services they want. Thus, i-mediators should participate in the common understanding of personal authorization attributes, too.

Inspecting the problem more thoroughly, we can hardly assume that autonomous entities in a worldwide information infrastructure can reach a common understanding about free properties implicitly. To master this challenge, we employ our approach of f-mediation that helps to interpret a free property as a bound property and to arrange the corresponding conversion. A f-mediator is then acting both as a verifier of free properties and as a delegatee and a grantor of a bound property on behalf of an i-mediator which is the owner of an information service, as visualized by Figure 4.9. In the basic design of i-mediation, the sources themselves still took the burden of this task.

In order to focus on the implementation of the basic hybrid PKI concepts and to keep the demonstrator manageable, we prefer to implement a quite simple i-

mediation scenario as part of the hybrid mediation scenario. For this purpose, we limited the functionality of an i-mediator and data sources to an owner's functionality as discussed in Section 4.4 and 4.5.

For the simulation of the hybrid mediation scenario, we have implemented software agents of four kinds according to the responsibilities of the entities involved in an instance of the hybrid PKI model:

- *Trusted authority agents* representing issuers of administrative properties and free properties as trusted authorities and licencees;
- *F-Mediator agents* representing verifiers of free properties and grantors of bound properties as delegatees;
- *User agents* representing holders of free properties and grantees of bound properties;
- *I-Mediator agents* and *data source agents* representing grantors and verifiers of bound properties as delegators and owners.

During our test runnings, which are reported in [50], we could successfully prove the feasibility of the key concepts of the hybrid model presented in Sections 4.5.1 and 4.5.2. It is important to note that the demonstrator is extensible and flexible enough to simulate other distributed applications like for example secure distribution of roles (see Section 5.2).

8.2. The Full Architecture

The motivation to elaborate the hybrid PKI model has not only originated from the conceptual challenge arising due to the incompatible security domains of i-mediation participants, but also due to a corresponding implementation task. Both the challenge and the implementation task are related to our view that all entities are highly autonomously cooperating, and accordingly all entities are implemented as software agents which base their communication on KQML [38, 46, 86] and their data exchanges on CORBA [67].

Figure 4.9 visualizes the interactions between the involved entities in an instance of the hybrid PKI model. Based on these interactions and on our view that all entities are highly autonomously cooperating software agents we primarily focused on two aspects in order to implement the intended demonstrator.

The first aspect is related to the KQML-based communication between the involved agents. The agents communicate by sending certain kinds of messages, called *performatives*, to each other. Considering the interactions involving among agents in an instance of the hybrid model, we need KQML performatives:

- to implement an instance of the model of the trusted authorities and licencing (as shown in the upper part of Figure 4.9);
- to implement an instance of the model of the owners and delegation (as shown in the lower part of Figure 4.9);
- to implement the conversion of free properties into bound properties (as shown in the middle part of Figure 4.9);
- to implement the access requests (i.e. which are directed to the owners) done by the grantees of bound properties (as shown on the left side of the lower part of Figure 4.9).

The second aspect deals with the implementation task with respect to software agents involved. The implementation task concerns the software agents for secure hybrid mediation. For any singular i-mediation request, the specific entities directly involved perform a fixed role as client, i-mediator or source, respectively, and those entities (i.e. which are involved in corresponding f-mediation) which possibly indirectly contribute perform a fixed role as (trustee or licensee/licensor or issuer) or (owner or delegatee/delegator or grantor), respectively. In general, however, all entities should be able to act in any of these roles during their lifetime. Thus, for the implementation of our demonstrator, we need a *core functionality* to be made available for all agents. In particular, all agents should be enabled to deal with free and bound properties and to convert the former into the latter. The required core functionality is provided by our *agent core*, the structure of which is presented in Section 8.2.2.

8.2.1. Extensions to KQML Performatives

In the original version of the KQML [38], security issues were not taken into consideration. The works in [86] and [46] made some changes with respect to secure communications and PKI related communications. We followed and extended the approaches of [86, 46] which were incomplete, especially since they do not fully satisfy the requirements of our hybrid PKI model. In order to implement KQML-based hybrid PKI, we proposed a new KQML ontology and recasted some performatives from [86, 46] and added new ones.

The new ontology is called secure mediation PKI, *smpki* for short, and enables the agents to know that the KQML performative they received concerns interactions involving in an instance of the hybrid PKI model. We use XML as encoding format for the data communicated through performatives. The recasted performatives are presented below.

- **applyDocument**[1]
  This performative is used by a *user agent* for its interactions with trusted authority agents, *f*-mediator agents, *i*-mediator agents and data source agents. In order to apply for a free property-certificate from a trusted authority agent and a bound property-credential from other agents listed above a user agent sends the following performative in the KQML message to the corresponding agents:

  ```
  applyDocument   :language XML
                  :ontology smpki
                  :content <requested properties and public
                  verification key>
                  [:certificateChains <free property-certificates
                  and corresponding licences>]
  ```

  The content of `content` is a signed request, the structure of which is quite similar to a digital document's structure (see Section 4.2.1). A signed request

[1]This performative is called `applyCredential` in [50]. In this thesis we would prefer to call it `applyDocument`. However, the usage of the performative is the same as in [50]. This performative has a dual usage. It is used for applying free property-certificates as well as for applying bound property-credentials.

includes the *requested properties* and the *public verification key* of the applying user agent.

In the case that a user agent uses this performative for applying free property-certificates, the *requested properties* might contain a set of free properties which the user agent previously received for inspection from the contacted trusted authority agent, i.e., by employing the standard KQML performative `tell`. In this case, the set of included free properties states the properties which are wished to be certified by the corresponding trusted authority agent as issuer.

If the performative `applyDocument` is used by a user agent to apply bound property-credentials, the *requested properties* might contain a set of bound properties which the user agent previously obtained for inspection from the contacted agent (e.g. f-mediator agent, i-medaitor agent, etc.). Again, for this purpose, the user agent employs the standard KQML performative `tell`. In this case, the set of included bound properties states the properties which are wished to be granted by the corresponding agent as grantor.

In each case above, the *public verification key* included in the signed request is used by the challenge-response protocol (i.e. which is used for authentication) applied by the contacted agent.

The parameter `certificateChains` is used only when applying for a bound property-credential. This parameter contains a set of certificate chains, the "main documents" of which include the set of free properties which are wished to be converted into bound properties. The authenticity of these chains have to be verified and validated (i.e. by the contacted agent) before granting bound property-credentials (see Section 4.4).

- **authChallenge** and **authResponse**
  The performative `authChallenge` is used by the agents acting in the roles of issuers or grantors. Before issuing a free-property certificate or granting a bound property-credential, the issuer or grantor, respectively, has to *challenge*[2]

[2] In some cases, it might be sufficient for a grantor to evaluate solely the certificate chains sent by a user agent before granting a bound property-credential, since the grantor may only want to gain assurance, whether the property encoded in the "main document" of a certificate chain is bound to the public key included in this document. In such cases, a grantor might not need to apply a challenge-response protocol.

the claiming user to prove that she holds the matching private key. In our demonstrator, the proof is accomplished by an appropriate *response* which is generated with the matching private key.

An issuer or a grantor sends the following performative in a KQML message to the user agent acting on behalf of the claiming user:

```
authChallenge  :language XML
               :ontology smpki
               :content <nonce>
```

The content of `content` in the `authChallenge` performative contains a randomly generated string which is to be signed by the claiming user. The user signs the string received and sends the following performative including the signed string to the corresponding agent:

```
authResponse  :language XML
              :ontology smpki
              :content <signed nonce>
```

- **issueCredential**
  This performative is used by the agents which are authorized to issue a certificate or a credential. Each authorized agent can send the following performative in the KQML message to other agents which have previously applied for a certificate or a credential by using the `applyDocument` performative:

  ```
  issueCredential  :language XML
                   :ontology smpki
                   :content <issued certificate or granted
                   credential with the corresponding chain of
                   supporting documents>
                   [:propertyDefinitions <definitions related to
                   the certified properties>]
  ```

  The parameter `content` contains the issued certificate or granted credential with the corresponding chains of supporting documents.

  The parameter `propertyDefinitions` is only used when issuing licence-certificates and granting delegation-credentials. In the former case, this parameter contains the properties defined by a licensor and which are to be used by its licensees when issuing further licence-certificates or free-property certificates. In the latter case, the parameter `propertyDefinitions` contains the properties and the implication rules related to these properties. The implication rules are used by the agents granting bound property-credentials (see Section 4.5.1). An implication rule[3] states a conjunction of personal authorization attributes required to issue a bound authorization attribute. The parameter `propertyDefinitions` is not included in the performative `issueCredential` when this performative is used to grant a bound property-credential by a f-mediator.

It is assumed that the properties of the requesting users, each of which is represented by a user agent in the virtual world, have been checked in an off-line manner by the trusted authorities or their representatives before issuing certificates or granting credentials to the corresponding user agents acting on behalf of their users.

In addition to using the recasted performatives and standard KQML performatives, we designed the following new performative for our demonstrator:

- **reduceCredential**
  As noted in Section 8.1, we limited the functionality of an i-mediator agent and data source agents to the functionality of an owner, as discussed in Sections 4.4 and 4.5. More precisely, the functionality of an i-mediator agent and data source agents is to handle the incoming `reduceCredential` performatives. This performative is designed to be used by user agents in order to apply for a reduced credential (see Section 4.4). A user agent may send the following performative in the KQML message to an i-mediator:

```
reduceCredential  :language XML
                  :ontology smpki
                  :content <chain of credentials>
```

  The parameter `content` contains a chain of credentials which has been previously obtained from a f-mediator or directly from an i-mediator or a data source agent.

[3]The work in [50] uses the term *issuing conditions* instead of implication rules.

In the case that a bound property-credential has been obtained from an i-mediator or a data source agent, a credential chain consists of solely a bound property-credential. Otherwise, a chain consists of delegation credentials as supporting documents. If the origin of the included chain is the i-mediator itself, then the i-mediator executes the credential reduction task immediately. It might be the case that an i-mediator receives a credential chain, the origin of which is not the i-mediator itself, but a data source agent connected with the i-mediator for cooperation. In such a case, the credential chain has to be verified by the corresponding data source agent. Thus, the i-mediator forwards the received chain to the corresponding data source agent. For this purpose, the i-mediator performs the following steps.

The i-mediator generates a new `reduceCredential` performative by putting the chain of credentials (i.e. which was included in the previous `reduceCreden-tial` performative sent to the i-mediator by the user agent) to the `content` of this new performative. Then the i-mediator sends the newly created performative in a KQML message to the corresponding data source agent. In the case of a positive evaluation of the received chain (see Section 4.2.4), the data source agent reduces the chain and answers by sending an `issueCredential` performative including the reduced credential. The i-mediator generates a new `issueCredential` performative including the reduced credential and sends the new performative to the user agent. In the case of a negative chain evaluation conducted by the data source agent, the standard KQML performative `sorry` is used either by data source agent or i-mediator agent to acknowledge the user agent.

There are two aspects which must be noted with respect to reducing a credential chain. The first aspect concerns the obligations in regard to granting bound property-credentials. It might be the case that an i-mediator agent or a data source agent may request the f-mediator (i.e. which is the grantor of a bound property-credential) to show the free-property certificates and the corresponding licence-certificates (i.e. which have been proved by the holder of free-properties and grantee of bound-properties) which must have been verified by this f-mediator before granting the bound property-credential. With this inquiry an i-mediator agent or a data source agent may aim at checking whether the f-mediator followed good practice before granting a bound property-credential. This situation requires that some obligations with respect to misusing delegated authorities are to be encoded in corresponding certification policies approved by the involved i-mediators and f-mediators. We handled this challenge in our demonstrator pragmatically. Before reducing a credential chain, an i-mediator agent or a data source agent first requests the corresponding certificates of the user agent by sending a performative `ask_all` to the corresponding f-mediator agent. The content of `content` in the performative is the bound property-credential granted by this f-mediator agent. Upon receiving such a request, the contacted f-mediator agent finds out the corresponding certificates and sends these by employing a `tell` performative.

The second aspect concerns authenticating user agents. It is important to observe that in the underlying scenario, the user agent obtains only a reduced credential granted in the positive case. There is no further service (i.e. except reducing a chain of credentials into a credential) triggered by the reduction process. Based on this fact, a data source agent does need to apply a challenge-response method for authenticating the user agent in the situation described above. Similarly, there was also no need to apply a challenge-response method in the specific i-mediation described in Chapter 6, as the sources deliver answers in *encrypted form* to the i-mediator.

With the KQML performatives introduced above we have successfully simulated the interactions among agents in the hybrid mediation scenario presented in Section 8.1. For other interactions (e.g. importing property definitions, requesting certificates from f-mediators) pertinent to our demonstrator, we used the standard KQML performatives `ask_all` and `tell` [38].

8.2.2. The Agent Core

After having introduced the need to develop an agent core, we can now present the structure of the agent core which constitutes a base software library serving for the construction of the specific agents listed in Section 8.1.

Figure 8.1 visualizes the structure of the agent core and the data flow among the components of the agent core. The agent core consists of five main modules: agent security module, agent functional module, agent human interface, agent communication interface and the agent PKI framework.

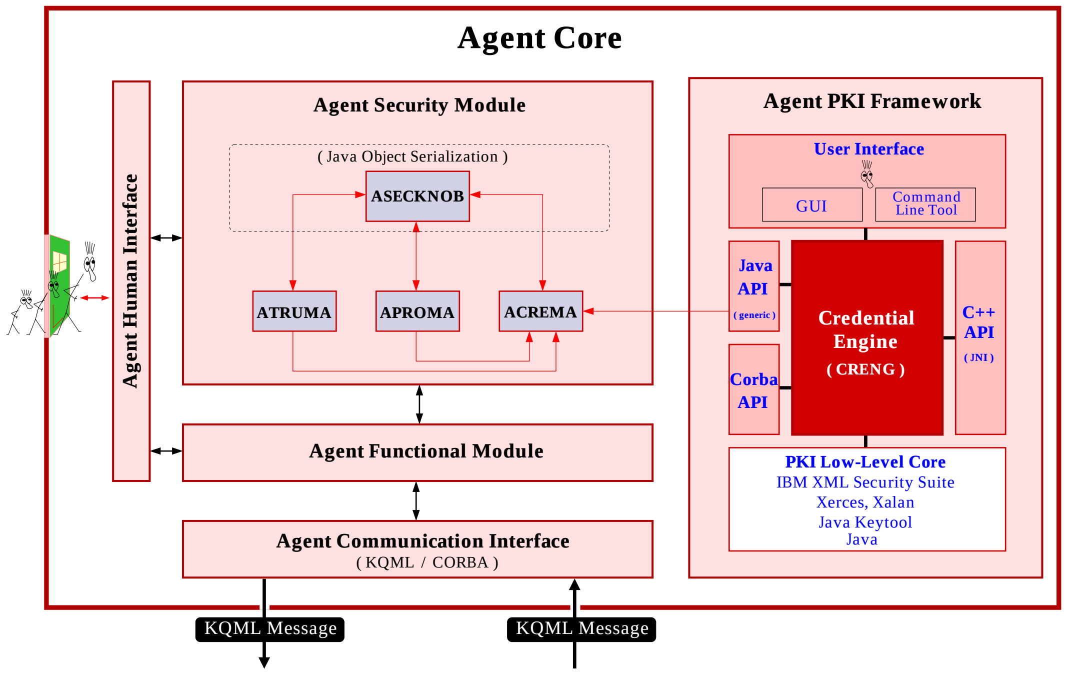


Figure 8.1.: Structure of the agent core

The *agent security module* has four components, the agent security knowledge base, ASECKNOB, the agent credential manager ACREMA, the agent property manager APROMA, and the agent trust manager ATRUMA.

The agent security knowledge base ASECKNOB maintains the following parts:

- The *property database* that is generated from the property definitions which are either locally defined by the administrator of an agent or imported from other agents.

- A *trust relationships database* that contains entries, each of which is related to a (trusted) agent and consists of the name, the public encryption key and the public verification key of the trusted agent. The entry also specifies the kind of the trust relationship (e.g. user agent, trusted authority, trusted f-mediator) regarding the agent named in the entry.

- The *certificates* and *credentials* including licence-certificates, free property-certificates, delegation-credentials, and bound property-credentials.
- A (Horn clause) *rule base* that specify implications among properties which are invoked when free properties are converted to bound properties.

The agent credential manager ACREMA is programmed to perform the following tasks:

- Issuing the licence-certificates, free property-certificates and granting bound property-credentials and delegation-credentials;
- Inserting the certificates and the credentials and the corresponding chains into ASECKNOB;
- Retrieving the certificates and the credentials and the corresponding chains from ASECKNOB;
- Evaluating the chains of certificates and credentials;
- Verifying the authenticity and validity of submitted certificates and credentials;
- Implementing the content management operations for certificates and credentials including extracting the free properties and bound properties from certificates and credentials, respectively;
- Implementing the operations needed to manage the keystore including fetching the public keys from the Java's keystore;
- Implementing the utilized challenge-response method for authentication.

In order to evaluate and to build the chains of certificates and credentials as well as to sign and to verify the individual certificates and credentials, the ACREMA makes use of the simple service calls of the CRENG module that belongs to the agent PKI framework.

The agent property manager APROMA implements the classes and functions which are responsible for the management and storage of property definitions. Furthermore, APROMA implements the property conversion policy used by corresponding agents in order to determine whether a given set of free properties qualifies for some bound properties. APROMA also provides methods for importing and exporting property definitions between agents.

The agent trust manager ATRUMA implements the operations needed to store and to retrieve the information (e.g. public keys) about the trusted agents.

The *agent functional module* implements classes and functions to process and evaluate the content of the KQML performatives. The functional module is a kind of scheduler analysing the incoming KQML performatives and scheduling the protocol steps to be executed. For this purpose the task control unit of the functional module coordinates the required function calls relating to the different components of the security module. What the content of a KQML performative expresses depends on the specific agent in which the performative is implemented.

The *agent human interface* is designed as an interface for the administrators of the agents to use and set up the corresponding agents. Depending on the kind of agent, the respective administrator is provided to perform a selected set of tasks through an agent specific graphical user interface. Below we list the main tasks which can be performed using the graphical user interfaces of the implemented agents. More precisely, through these interfaces the administrators can:

- maintain the property definitions including the insertion, deletion and update of property definitions;
- maintain the trust relationships database by inserting new entries, and deleting and updating the existing entries;
- import property definitions from other agents;
- issue licence-certificates and free property-certificates;
- grant delegation-credentials;
- grant bound property-credentials;
- visualize the received certificates and credentials in XML format.

The *agent communication interface* implements classes and functions for sending and the reception of CORBA messages which are used to communicate performatives.

The agent core is implemented in Java [83] and on a Solaris platform. The agent core uses the agent library *libAgent*, which has been developed within the framework of MMM project [8] in our working group. The demonstrator employs XML-encoded [29, 74, 69, 5] certificates and credentials, handling of which is provided by our agent PKI framework.

With the help of the libAgent's development stage, we could focus on the application specific details of our agents, since the libAgent smoothly hides the CORBA related development complexity.

8.2.3. A Selected Operating Sequence: Acquiring Credentials

In this section we present the operation seqence of an `applyDocument` performative sent from a user agent to a *f*-mediator agent (see Section 8.2.1). The performative is used by the user agent to apply for bound property-credentials. We assume that the user agent has already collected his free-property certificates from some trusted authority agents and received the bound properties for inspection (i.e. by using the standard KQML performative `tell`) from the *f*-mediator agent before applying for a bound property-credential. After the task control unit of the functional module has received the performative and conducted a challenge-response procedure successfully, basically the following steps are performed for the case that all security checks are positively evaluated (otherwise appropriate exceptions are executed):

1. The task control unit extracts the contents of the parameters from the performative and inserts them into ASECKNOB by calling the appropriate persistence functions of APROMA and ATRUMA.

2. The task control unit sends the signed request to ACREMA.

3. ACREMA verifies the authenticity of the signed request included in the `content`. For this purpose ACREMA employs the appropriate function calls of CRENG.

4. ACREMA extracts the requested bound properties and the public verification key of the user agent from the signed request.

5. ACREMA asks APROMA whether the requested bound properties are locally known.

6. After verifying the authenticity and the validity of the free property-certificates and the corresponding licence-certificates, ACREMA extracts the free properties from the verified certificates. The set of free properties are forwarded to APROMA.

7. APROMA checks whether the included set of free properties “qualifies for” bound properties requested.

8. The result of checks are forwarded to the agent human interface and shown by this interface to the f-mediator administrator.

9. After inspecting the shown values, the administrator generates a bound property-credential encoded in XML. The encoded XML document is forwarded to ACREMA.

10. ACREMA signs the generated XML document by using CRENG. The signed document now constitutes the issued bound property-credential. ACREMA adds the corresponding delegation credentials to the issued bound property-credential.

11. Finally, the task control unit prepares an `issueCredential` performative in which the issued bound property-credential and delegation credentials are encoded. The performative is sent in a KQML message to the requesting user agent.

8.3. Contributions

In this chapter we made the following contributions:

- We demonstrated a hybrid mediation scenario underlying the hybrid PKI demonstrator.

- We presented the extensions to KQML performatives which are used by the agent interactions that are involved in an instance of the hybrid PKI model.

- We exhibited the high-level architecture of the agent core which is used to construct specific agents as needed by an instance of the hybrid PKI model.

- We presented the main features of the components of the agent core.
- Finally, we demonstrated an elaborated control flow among the functional and security components for a selected operating sequence.

9. The Security Module of the MMM Agent

In this section we provide more details on how *secure i-mediation*, as explained in Section 6.4.3, is actually implemented as part of the MMM. The security architecture of the MMM was visualized by Figure 6.5 (see Section 6.4).

The demonstrator of the security module was implemented to prove the feasibility of the main ideas. The major focus of the demonstrator was the implementation of two algorithms:

- An algorithm to verify the accreditation-certificates (see Section 4.3.2) submitted by a client and to implement the operations related to the management of personal authorization attributes extracted from these accreditation-certificates.
- A validation algorithm to check whether a given request is permitted on the basis of personal authorization attributes (see Sections 6.4.3.3 and 6.4.3.4).

When we implemented the demonstrator, no real external data source was available in the MMM environment. Hence, the demonstrator does not include an implementation of the *answer encryption policy* for *i*-mediation as explained in Section 6.4.3.5. Furthermore, at the time of writing this thesis, the functionality of dynamically generating external views (see Section 6.4.3.3) and the functionality of the (Horn clause) rule base (see Sections 6.4.1 and 9.1.1) have been implemented, and the feasibility of the corresponding key concepts have been successfully proved in an off-line manner. Hence, these functionalities could not have been proved in the test runnings of the demonstrator. Otherwise, the demonstrator implements all features (see Section 6.4) which were introduced so far. During our test runnings in which a user agent, an *i*-mediator agent and two data source agents were involved, we could successfully demonstrate the secure *i*-mediation phases (i.e. with the limitations stated above), as presented in Section 6.4.3.

The demonstrator was implemented in C++ and used the object-oriented database system O2 [11]. The MMM itself employs CORBA [67] for data exchanges, KQML [38] for agent communications, and ODL/OQL [24] for schema declarations and query expressions.

The demonstrator assumes the existence of an appropriate access control layer for the underlying object-oriented database management system. Unfortunately, this layer is not provided by O2.

The demonstrator employs the credential engine CRENG of the agent PKI framework for the verification of the certificates and credentials encoded in XML.

When implementing the security module, the most important design objectives we had in mind were:

- The security module shall be provided with appropriate tools to support an extended design of i-mediation employing the hybrid PKI model (see Section 4). It is important to note that for the present demonstrator of the security module, we only need PKI services in regard to handling accreditation-certificates and licence-certificates, as the underlying design of i-mediation (see Section 6.3) is based on the model of trusted authorities and licencing (see Section 4.3). To achieve this objective regarding the extended design of i-mediation, we equiped the credential manager CREMA (see Section 9.1.1) of the security module with appropriate mechanisms needed for issuing and verifying certificates (see Section 4.3.2) as well as credentials (see Section 4.4.2).

- The authorization decision engine MAIDEN (see Section 9.1.1) of the demonstrator shall be unaware of the employed PKI model. We achieved this by implementing separate components responsible for the management of certificates and credentials, and regulating access, respectively.

- The credential manager CREMA shall be unaware of the underlying encoding formats, and of the signature algorithms used to sign the incoming certificates and credentials. To achieve this objective, we separated the MMM-specific PKI tasks (see Sections 9.2.1.2 and 9.2.3) from the low-level PKI services with respect to handling digital signatures. Due to the separate implementation of these functionalities, replacing the XML encoded documents with other digital documents having different encoding formats (e.g. ASN.1 used for X.509

certificates or s-expressions used for SPKI authorization certificates) does not cause any modification in the core of the security module. Only the low-level PKI core of the credential engine CRENG needs to be extended to handle new formats and cryptographic signature algorithms.

The design is presented at the object modelling level using UML. The UML classes are not discussed in detail. Only method names are given instead of complete interfaces. Classes which appear as gray boxes are presented in another figure.

The demonstrator has an elaborated design. For the sake of simple exposition, we only present the major classes and methods.

The rest of this chapter is organized as follows. Section 9.1 describes the main components of the security module and sketches the interactions between the functional and security components for a request phase (see Section 6.4.3 and Figure 6.5). Section 9.2 refines the core of the secure MMM and presents the collaboration among the runtime objects of the functional and security components and the modifications made in the existing MMM prototype, in order to implant the security module into the MMM prototype.

9.1. The Outline of the Security Module

In this section we outline the main components of the security module and demonstrate a typical control flow among the functional and security components for a request phase (see Section 6.4.3 and Figure 6.5). This sketch is intended to demonstrate the subtle interactions between functional and security considerations found for i-mediation.

9.1.1. The Main Security Components

The security module of the MMM has three main components, the security knowledge base SECKNOB, the credential manager CREMA, and the i-mediator authorization decision engine MAIDEN (see Figure 6.5).

The security knowledge base SECKNOB maintains the following parts:

- The *authorization databases* (see Section 6.4.1) that are generated from the annotated declarations (see Section 6.4.2) of the application schema, the twin schema and the representations of the external source schemas (see Sections 6.4.3.1 and 6.4.3.2).

- The *accreditation-certificates* (see Section 6.3.2) submitted by a client.
- The *personal authorization attributes* (see Section 6.4.1) extracted from the accreditation-certificates together with links associating them with the accreditation-certificates they are extracted from.
- The *keystore access data* that contains the passphrases needed to access the MMM's Java keystore and the private key (i.e. which belongs to MMM agent) stored in the same keystore (see Section 7.1).
- A (Horn clause) *rule base* that specifies implications among authorization expressions which are invoked when an instance of the relationship "*qualifies for*" between authorization expressions (see Section 6.4.1) must be decided. The rule base is intended to take advantage of "real world" properties of personal authorization attributes (see Section 6.4.1), 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 [13, 52].

The credential manager CREMA is responsible for the verification of the authenticity and validity of submitted accreditation-certificates. Moreover, it extracts personal authorization attributes from accreditation-certificates, inserts accreditation-certificates, their personal authorization attributes and the pertinent links into SEC-KNOB, and determines accreditation-certificates and the related licence-certificates to be included to subqueries. For the actual verification of the authenticity and validity of submitted certificates and extraction of personal authorization attributes, the CREMA makes use of the simple service calls of the credential engine CRENG of the agent PKI framework (see Section 7).

The *i*-mediator authorization decision engine MAIDEN implements the *authorization policies for i-mediation*, both for *schemas* and for *instances* (see Sections 6.4.3.3 and 6.4.3.4), and the underlying *pure* authorization policy (see Section 6.4.1) together with its basic relationship "*qualifies for*" among authorization expressions.

The object-level refinement of each component outlined above is presented in Section 9.2.

9.1.2. Query Evaluation with Accreditation-Certificates

In this section we present a control flow among the functional [9] and security components (see Section 6.4 and Figure 6.5). We assume that exactly one client request has to be handled. Thus, we ignore all identifications and references needed to appropriately relate actions and data belonging to a single global request. After the query rewriter has received the request, basically the following steps are performed for the case that all security checks are positively evaluated (otherwise appropriate error-handling routines are invoked):

1. The query rewriter sends the included accreditation-certificates with the associated licence-certificates (see Section 4.3.2) to CREMA.

2. CREMA evaluates the accreditation-certificate chains, extracts the personal authorization attributes and inserts the accreditation-certificate chains, their personal authorization attributes and the pertinent links into SECKNOB.

3. The query rewriter asks MAIDEN whether the request is allowed according to the *pure* authorization policy applied to the annotated *application schema*.

4. MAIDEN decides allowance using SECKNOB.

5. The query rewriter analyzes and decomposes the global query. A set of (functionally equivalent) query plans is forwarded to the optimizer.

6. The optimizer determines an optimal query plan and delivers it to MAIDEN for the final enforcement of the *instance authorization policy for i-mediation.*

7. MAIDEN confirms that all requirements of the instance authorization policy for *i*-mediation are met, and then forwards the optimal query plan to the query evaluator.

8. Finally, the query evaluator gathers the required accreditation-certificate chains with the help of CREMA, produces the subrequests to the sources and forwards them to the twin source and the wrappers, respectively.

9.2. Refining the Secure MMM Core

In this section we show the dependencies between the respective runtime objects related to the components involved in the query evaluation presented above.

9.2.1. The Security Knowledge Base

The implementation of the SECKNOB uses the object-oriented database environment of O2 [11]. In the following sections we give more details on how some parts of the SECKNOB are implemented.

9.2.1.1. The Authorization Database

In order to generate authorizations from the annotated declarations of the application schema, the twin schema, and the representations of the external source schemas (see Sections 6.4.3.1 and 6.4.3.2), we have done implementations concerning the following tasks:

- We defined an annoted ODL grammar, in order to make the existing ODL parser (i.e. which has been developed for the MMM agent) capable to handle annotations (see Section 6.4.2). Accordingly, we extended the ODL parser.
- We declared an authorization database schema (see Figure 9.1). An instance of the authorization database schema represents access authorizations related to the schema annotations and instance annotations (see Section 6.4.2) of an application schema, a twin schema or the representation of an external source schema. We implemented the authorization database schema in O2C language [11].
- We developed a runtime module to handle C++ objects (i.e. which constitute the output of ODL parsing process) representing the access authorizations. The runtime module instantiates the authorization database schema for storing the translated C++ objects into SECKNOB and makes sure that the corresponding instantiation is stored persistently in the O2.

The left part of Figure 9.1 visualizes a class model representing the declared authorization database schema. The MMM maintains an elaborated meta schema

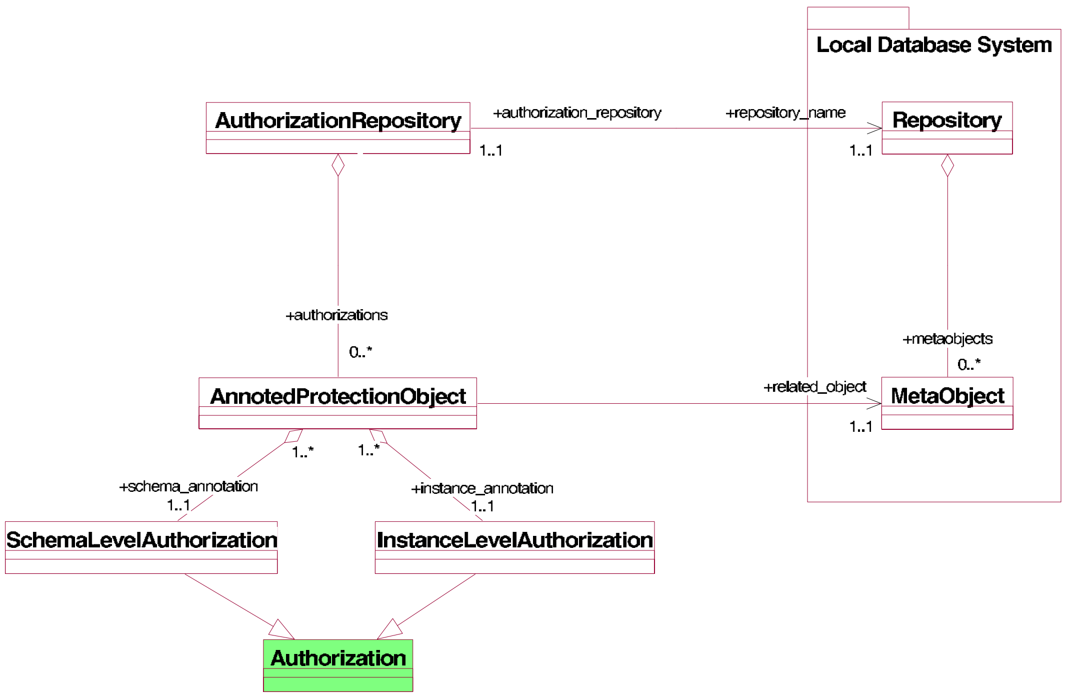


Figure 9.1.: A class model representing the authorization database schema

based on the ODMG proposal. The right part of Figure 9.1 depicts only those meta schema classes, being directly referred to by the authorization database schema. A detailed presentation of the meta schema is beyond the scope of this thesis. Coarsely speaking, an object of the class `Repository` stores the declarations of an imported schema. Schema declarations (e.g. class, attribute, extent, etc.) are represented by the objects of the class `MetaObject`.

For each `Repository` object there exists an object of the class `Authorization-Repository`. The class `AuthorizationRepository` represents the authorizations that are generated from the annoted declarations of an imported schema. The annotations related to a schema declaration are represented by the class `Annoted-ProtectionObject`. Each object of the class `AnnotedProtectionObject` contains two `Authorization` objects, each of which is generated from the schema annotation and the instance annotation of a schema declaration. Each `AnnotedProtection-Object` object is associated with a meta object for which the corresponding authorizations are maintained. The schema annotations are represented by the class

SchemaLevelAuthorization and the instance annotations are represented by the class InstanceLevelAuthorization. A class model representing the structure of the class Authorization is already visualized by Figure 6.6 (see Section 6.4.1).

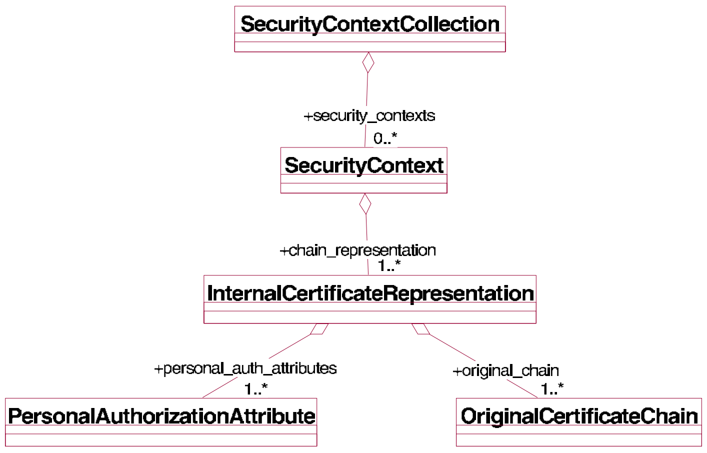

SecurityContextCollection
+security_contexts
0..*
SecurityContext
+chain_representation
1..*
InternalCertificateRepresentation
+personal_auth_attributes
1..*
+original_chain
1..*
PersonalAuthorizationAttribute
OriginalCertificateChain

Figure 9.2.: A class model representing a database schema for security contexts

9.2.1.2. The Security Context

Figure 9.2 visualizes another part of the SECKNOB, which maintains the certificates and the personal authorization attributes submitted by a client (see Section 6.3.2). The class InternalCertificateRepresentation constitutes the main class of the class model visualized by Figure 9.2. An object of this class stores the original certificate chain submitted by a client and the personal authorization attributes extracted from the accreditation-certificate included in the chain. A client's request may contain a set of certificate chains. The internal representations of the submitted certificate chains are stored in an object of the class SecurityContext. A SecurityContext object is identified by a unique *query identifier* (i.e. determined by the functional core of the MMM), when a client's request is received by the MMM. A query identifier is used to refer to a client's request, including a global query and the client's certificate chains. An object of the class SecurityContextCollection is a container in which the SecurityContext objects stored.

9.2.2. The Functional Core

In this section we outline the functional core of the MMM agent [9]. We emphasize only the parts of the functional core, which are needed to explain the functionality of the security module in the MMM and how the security module has been implanted into the MMM agent. The functional core of the MMM consists of three major parts (see Figure 9.3).

- a controlling unit managing program flow and interaction with the user
- the actual working part doing heavy work like optimizing and evaluating queries
- the local database containing schemas, proxy states, and access information

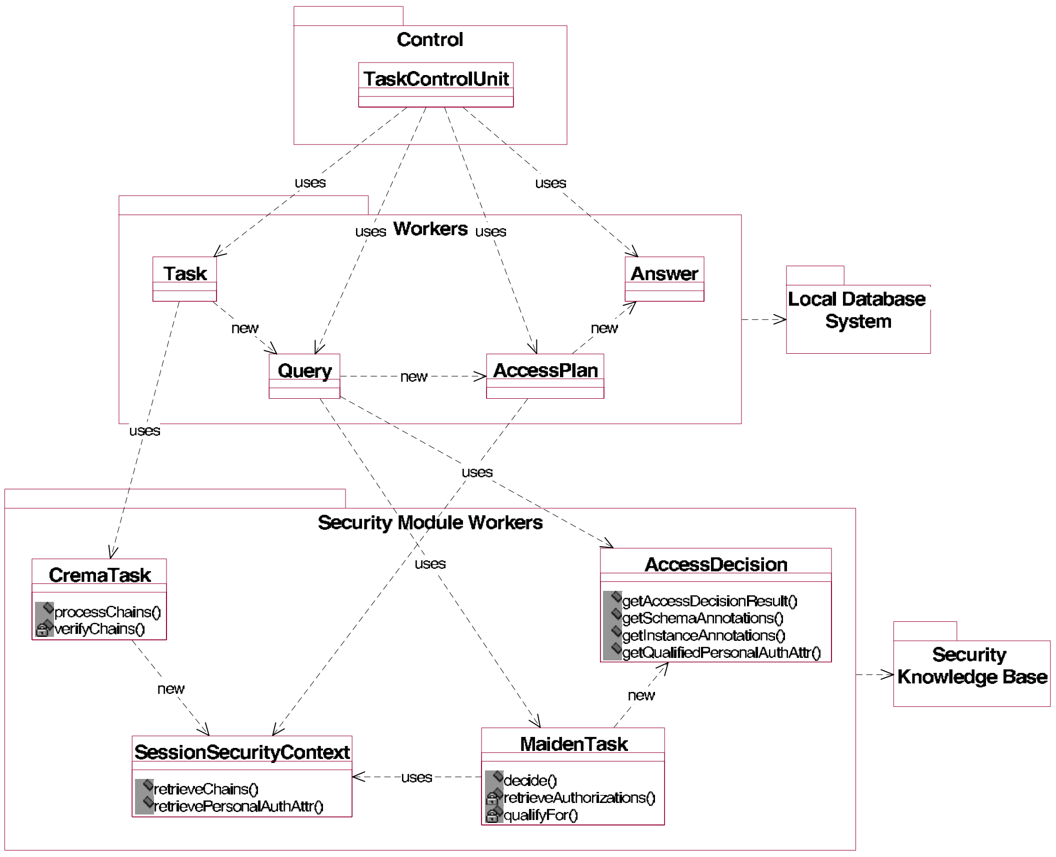


Figure 9.3.: A class model representing the dependencies between the worker objects of the secure MMM core

The MMM's work is directed by an object of class `TaskControlUnit`, which is the point of access for user agents or other i-mediators. A `TaskControlUnit` object controls the mediating process by invoking a parser, an optimizer, and a query evaluator.

The working part consists of four major classes: `Task`, `Query`, `AccessPlan` and `Answer`. A `Task` object is the one submitted by the user agent. It contains the input given by the user. The input contains an OQL query as a string and the certificate chains (i.e. submitted by the client) as a set of strings[1]. The class `Query` constitutes a representation of an OQL query. The main functional task of a query representation is to decompose itself in order to distribute subqueries among external sources. Objects of the class `AccessPlan` contains a detailed description of how to distribute and evaluate a query with respect to available external sources. An `Answer` object only contains an identifier representing the symbolic name of the answer, which is actually stored in the local database. The identifier can be referred to by other subqueries or by the client or user agent, respectively.

The local database system contains a meta schema, i.e. a schema which can be instantiated to describe an application schema and several source schemas.

9.2.3. Functional Core Objects Meets Security Module Objects

In Section 9.1.2, we demonstrated the collaboration among the functional and security components of the MMM core. In this section we demonstrate the dependencies between the objects corresponding to these components.

The major functional task of the class `CremaTask` is to handle the certificate chains submitted by a client. For this task to be accomplished, a `Task` object invokes the method `CremaTask::processChains` with a set of certificate chains submitted by a client and a unique query identifier assigned to the client's request (see Section 9.2.1.2) as arguments. The method `CremaTask::processChains` employs `CremaTask::verifyChains` which makes use of the simple service calls of the credential engine CRENG (see Section 7) for the actual evaluation of client's accreditation-certificate chains. A call to the method `CremaTask::processChains` returns a `SessionSecurityContext` object that contains positively evaluated original certificate chains and the personal authorization attributes extracted from the

[1]Note that a certificate chain is represented as a XML document as explained in Section 7.2.

accreditation-certificates of each chain. A `SessionSecurityContext` object can be seen as an internal representation of the verified evidences of a client's eligibility. A `SessionSecurityContext` object is a runtime representative of the corresponding persistent `SecurityContext` object (see Section 9.2.1.2).

The main task of the class `MaidenTask` is to implement the algorithms corresponding to the *authorization policies for i-mediation* – both for *schemas* and for *instances* (see Sections 6.4.3.3 and 6.4.3.4) – and the underlying *pure* authorization policy (see Section 6.4.1) together with its basic relationship "*qualifies for*" among authorization expressions. An object of the class `MaidenTask` is used by a `Query` object by calling the method `CremaTask::decide`. The parameters needed to be passed to this method are `repository_object`, `repository_name` and `query_identifier`.

The `repository_object` contains the *absolute name* of the repository object which is stored in the repository determined by the value of the argument corresponding to the parameter `repository_name`. The `query_identifier` is used to determine against which `SessionSecurityContext` object a respective *authorization policy* should be evaluated.

In the following we present the performed steps and the respective method invocations done by the method `CremaTask::decide` to implement the *pure* authorization policy (see Section 6.4.1) together with its basic relationship "*qualifies for*" among authorization expressions:

1. Retrieve the *authorization expression* a defined for the given repository object from a given authorization repository stored in SECKNOB. The authorization expression a is defined over atomic personal authorization attributes which are in *disjunctive normal form*. For this step to be accomplished, the method `MaidenTask::retrieveAuthorizations` is employed.

2. Retrieve the *authorization expression* r, which is a conjunction of the personal authorization attributes, built from the client's accreditation-certificates. For this step, the method `SessionSecurityContext::retrievePersAuthAttr` is invoked.

3. Check whether there exists a conjuction c in the authorization expression a, for which authorization expression r qualifies. More precisely, this step searches for a conjunction c in the disjunction r which constitutes a subset of the conjuntion

r. In the positive case, the conjunction c found in r represents a client's qualified personal authorization attributes. The task of this step is accomplished by the method `CremaTask::qualifyFor`.

A call to the method `CremaTask::decide` returns an object of class `AccessDecision`, which is used by the object `Query` in order to generate (functionally equivalent) query plans. An `AccessDecision` object contains the decision of whether the request is permitted. In the positive case, an `AccessDecision` object contains the personal authorization attributes (i.e. represented by the conjunction c), on the basis of which the access decision was positively evaluated. In this case, an appropriate method of an `AccessPlan` object employs the method `SessionSecurityContext::retrieveChains` to gather the required accreditation-certificate chains (i.e. corresponding to the personal authorization attributes contained in the conjunction c) which are to be attached to the subqueries (see Section 9.1.2). In the negative case, an `AccessDecision` object contains the respective annotations which could be used to inform the client about the required personal authorization attributes for a positive evaluation of her request.

9.2.4. Implanting the Security Module into the MMM Agent

In order to test the functionality of the security module in the existing MMM-specific i-mediation environment, we made following modifications in the existing MMM prototype:

- The appropriate functions provided by the application programming interface of the security module have been implanted into the corresponding worker classes of the functional core, as presented in Sections 9.2.2 and 9.2.3.
- In order to be able to send a client's certificate chains from a user agent to the MMM agent, we designed a new KQML performative by making a minor modification to the existing standard KQML performative `ask_all`. The new performative is called `ask_all_secure` and includes now a new parameter `:certificateChains` in which the certificate chains of a client are encoded.
- We extended the user agent implemented as a Java application to support clients in managing their certificates. This includes presently reading the cer-

tificate chains from the file system and visualizing them to the clients for selection and submission to the MMM agent.

9.3. Contributions

In this chapter we presented the details on how *secure i-mediation* is actually implemented as part of the MMM. We made the following contributions:

- We outlined the security module of the MMM.
- We demonstrated a control flow among functional and security components of the security module.
- We then refined the security knowledge base of the security module.
- Afterwards, we presented the dependencies between the respective runtime objects related to the components which are involved in a typical query evaluation.
- Finally, we outlined a validation algorithm for checking whether a given request is permitted based on personal authorization attributes.

Part VI.

Summary

10. Related Work

Existing research approaches, which are related to the problems we tackled, are rooted in two research areas: secure mediation and certificate/credential-based access control in distributed information systems. These approaches are based on different motivations and requirements, which have shaped the corresponding solutions and had significant impact on their specific security architectures. In order to compare our solutions – related to the research areas stated above – with existing approaches, we first present the requirements corresponding to each of our solution, and then evaluate the existing approaches against these requirements.

The rest of this chapter is organized as follows: In Section 10.1, we compare our approach of secure i-mediation with other approaches. First, the characteristics of the i-mediation environment are presented and the requirements on security for i-mediation are inferred from these characteristics (Section 10.1.1). Second, existing approaches are discussed (Section 10.1.2), and third, these approaches are evaluated against the inferred security requirements (Section 10.1.3). In Section 10.2, we compare our approach of certificate/credential-based access control with an emphasis on PKI related requirements to other works. First, the PKI related requirements are outlined (Section 10.2.1). Existing solutions are then presented (Section 10.2.2). Finally, these requirements are evaluated against the outlined requirements (Section 10.2.3).

10.1. Secure *I*-Mediation

In the following sections we outline our approach of secure i-mediation, and then give a comparision of our approach to existing approaches.

10.1.1. Requirements on Security for *I*-Mediation

Before exhibiting the security requirements for i-mediation, we present the security-relevant aspects, as seen from our point of view. These aspects have shaped the requirements, the design, and the security architecture of our secure i-mediation approach (see Section 6). A detailed discussion of issues, which are handled in this section, was given in Section 6.2. We outline them as follows:

- There is no a priori organizational base for direct mutual trust between users, i-mediators and sources.
- The clients are unwilling to divulge their identities for private reasons and thus prefer to remain anonymous.
- For a resource owner, it is necessary to see evidences of a client's eligibility rather than to know *who* they are.
- The interoperability is stimulated by spontaneous clients.
- Clients and information sources are independent of mediators.
- Clients are independent of information sources, if there are several suppliers for their demand.
- Information sources exist in competitive, as well as in non-competitive relationships with other information sources.
- While information sources may have cooperation contracts with i-mediators, it can be assumed that spontaneous clients are unknown beforehand and cannot be registered with i-mediators in a static manner before queries can be satisfied.
- The group of information sources are not stable, due to the lack of organizational associations in mediated systems.
- I-Mediators should strive for tolerance with respect to information provider failures and offer services to ad-hoc clients which have not registered with the services beforehand.
- There is no useful assumption of a closed world in mediated systems due to their dynamics.

Based on the security-relevant aspects presented above, the following security requirements are inferred:

1. *Authenticity* should be based on *certified personal authorization attributes.* A client proves his eligibility to see a piece of information by a collection of his personal authorization attributes – which are evidences of his eligibility – and each autonomous source follows a security policy that is expressed in terms of personal authorization attributes.

2. *Confidentiality* should be based on *certified personal authorization attributes.* Any source should autonomously follow a security policy with respect to confidentiality which ensures that requested information is delivered only to that client which provided the inspected evidence.

3. The *authorization requirements* of the *i*-mediator and the sources are *conjunctively* combined: access to source data via the *i*-mediator is permitted iff it is allowed by *both* the *i*-mediator *and* the source. Accordingly, the security module of the *i*-mediator acts as a kind of *filter* placed in front of the sources.

4. The *i*-mediators may maintain their own data and grant authorizations with respect to their data.

5. *I*-Mediators and information sources should *grant* authorizations with respect to their data *autonomously.* There is no need for a coordinated security policy between *i*-mediators and information sources.

6. *I*-Mediators and information sources should *decide* on permitting requests *autonomously.*

7. *I*-Mediators should be able to *forward* submitted personal authorization attributes to the information sources.

8. The design of an *i*-mediator should be based on a top-down design paradigm that allows for a stable data description of integrated information at varying degrees of source fluctuation.

10.1.2. Existing Approaches

Security in interoperable information systems has been mostly investigated within federated database contexts, where the emphasis laid on resolving heterogeneity of access rights among the components. These works have been characterized by the fact that access restrictions are defined on a global federated schema requiring a redesign each time a local schema changes or is added. The work of Jonscher et al. [55, 54] describes a configurable access control system called Argos, which is able to enforce a variety of access control policies in the area of identity-based access control. In this approach the management of global vs. local identities and their authentication, and enforcement of different policies by the sources are addressed. Access requests are evaluated by so-called reference monitors. Each access request is forwarded to the responsible reference monitor for the object to be accessed. Each reference monitor encapsulates a security policy. Tari et al. [85] address the issue of identification and access control in the federated database system called DOK. The authors propose an object-oriented model called Unified Security Model aiming for the integration of existing access control models which could have been imposed on local components of a DOK application. Samarati et al. [28] address specification and bottom-up derivation of authorizations and consideration of administrative privileges. All these approaches are clearly inapplicable in our context.

For contributions to secure i-mediation see [23, 92, 93, 94, 27].

Subrahmanian et al. [23] provide a formal model of security in a mediated system, in which the information sources obey a mandatory security policy. This work does not address the problem of authentication. The authors state that different packages[1] participating in a mediated system may all use different security models and may all respond in different ways. In order for the mediator[2] to successfully access an external package, they assume that these local packages have a security wrapper associated with them. The requests by the mediator are first processed by this security wrapper. The security wrapper consists of a segment of code that implements a local security check policy. Each of the individual packages may enforce its own security policy. The mediator follows a global security policy and respects the security policies of the individual packages. According to the global policy, a global request only succeeds

[1] A *package* corresponds to an information source in the sense of this thesis.
[2] A *mediator* corresponds to an i-mediator.

if the mediator and all involved packages validate the request (or corresponding subrequests) as being permitted. In [23] the authors present two possible ways of handling security in a mediated system:

- *Mediator-enforced security:* The packages *trust* the mediator and allow the mediator to enforce the security policy of the packages. In this case, the appropriate security criteria must be encoded as rules within the mediator and these rules must be enforced by the mediator.

- *Package-enforced security:* The packages *do not trust* the mediator. In this case, the mediator serves two purposes:

  - The mediator uses packages that either allow mediator-enforced security or impose relatively few restrictions on security. Packages that enforce stringent security will only be used when other means of processing the query fails.

  - The mediator serves merely as a communication medium, conveying the security screening requests of the package to the user, and returning the user's responses to the package.

In this work the authors focus on the how the mediator handles package-enforced security.

The security techniques described in [23] have been implemented as a component of the *Heterogeneous Reasoning and Mediator System*, HERMES. When the HERMES is invoked, the system asks the user to identify herself and then determines the person's security clearence level from a database. At run-time, when the user attempts to execute a specific domain function, the system checks to see if the user is cleared to access the corresponding domain function.

Wiederhold et al. [92, 93, 94] present a security mediator regulating access to medical databases. This work involves access control and sanitization of answers before they are returned to the subject. The security mediator is administered by a security officer who is responsible for enforcing the medical institution's security policies. The major tasks of the mediator include translating security policies into a set of rules, regulating access to medical data, and informing the security officer when a rule violation is detected. In case of a rule violation, the security officer approves the query or a filtered set of query results. The medical databases do not have their own security mechanisms. The access to these databases is only regulated by the security mediator and the corresponding security officer. Hence, the security mediator can be seen as a kind of gateway between clients seeking information and medical databases. This approach employs a traditional authentication mechanism which is based on usernames and passwords. This secure i-mediation approach is clearly inapplicable in an open and interoperable i-mediation environment whose security requirements were discussed in Section 10.1.1.

Samarati et al. [27] propose a mediator-based solution to the problem of providing secure interoperation among independent information sources protected by mandatory policy restrictions. The approach presented in this work is some what similar to our approach of secure i-mediation. Each information source grants authorizations and bases access decisions autonomously. Access control is enforced at both the mediator and source levels. The mediator maintains a virtual application schema. In contrast to our approach, the application schema is defined in terms of the schemas of the local data sources, and the authorizations – in this case security levels – granted for the application schema are derived from the security levels assigned to the objects at the sources. As visualized by Figure 6.2 and discussed in Section 6.4.3.1, we followed a top-down design approach where the administrator of the i-mediator declares an application schema and grants authorizations for the application schema independent of the source schemas and the corresponding authorizations, respectively. The work in [27] employs security clearances rather than using user identities, in order to avoid the burden of dealing with identities. The authors do not address how the authenticity of the security clearences shown by the users are verified. The authors state that their approach could be enhanced by employing certificate/credential-based access control mechanisms as presented throughout this thesis.

10.1.3. Evaluation of Existing Approaches

The common feature, which our approach and existing approaches share, is that all of these approaches have drawn their inspirations from the same work presented in [91, 95] which proposes the concept of a *mediator*. However, the security requirements of the existing i-mediation solutions differ from those which we have outlined in

Section 10.1.1.

Existing solutions employ either identity-based or security clearance-based authentication and authorization approaches which appear to be less useful for i-mediation. Considering the dynamics and the security requirements of i-mediation participants, we argue that the employment of merely identity-based and security clearance-based authentication and authorization approaches is inadequate and ineffective for pervasive computing systems such as i-mediation. Secure i-mediation requires an approach to security based on the certified personal authorization attributes, as presented in Section 6.

10.2. Certificate/Credential-based Access Control

In the following sections we outline the requirements on certificate/credential-based access control with an emphasis on PKI related requirements. After discussing existing solutions, we evaluate them against the outlined security requirements.

10.2.1. Requirements on Certificate/Credential-based Access Control

In Section 10.1.1, we exposed the security requirements for i-mediation. We argue that many of these requirements are valid not only for i-mediation, but for many other applications as well. The following requirements are crucial:

i) A client should be represented by (one of) her public keys and characterized by the assigned properties. Avoiding identification is one of the necessary conditions for anonymity. For this purpose, in addition to obtaining certificates which encode the identifying properties of a client, it should be possible for a client to acquire certificates/credentials which encode her non-identifying properties (e.g. personal attributes) and to prove these non-identifying properties without revealing her identity.

ii) A trusted authority should assign properties to public keys.

iii) A server should follow a security policy that is expressed in terms of characterizing properties.

However, most of the distributed applications which are based on these requirements are faced with the following conceptual challenge. The participating entities mostly belong to autonomously operating domains and are not in a position to deal with the heterogeneity of property-related issues (see Section 5.2). From this conceptual challenge, the following security requirements are inferred:

iv) Clients should be assisted by a server to assemble appropriate properties within suitable digital documents, in order to receive the information services they want.

v) A server should be able to delegate the authority over evaluating a property defined in a foreign domain (i.e. which might be incompatible with the domain of the server) to an expert entity which should act on the behalf of the server and manage the complexity inherent in independently defined properties.

vi) An entity should be able to use one property to make inferences about another property.

vii) It should be possible for an entity to infer a property p based on the conjunction of other properties where such a conjunction specifies all the properties required to grant the property p.

viii) An entity should be able to infer a property based on the field value of another property.

10.2.2. Existing Approaches

Existing approaches can be classified into two main categories: PKI defining approaches and PKI applying approaches. In the following sections we present existing approaches and evaluate them against the security requirements stated above.

10.2.2.1. PKI Defining Approaches

As discussed in Section 4, existing approaches for defining a PKI are based either on trusted authorities with licencing or on owners with delegations.

The basic approach of widespread X.509 public-key infrastructure [4, 51] is based on trusted authorities and licencing. The key-oriented trust management systems

PolicyMaker [17, 18] and KeyNote [16] of Blaze et al. and the SPKI/SDSI approach of Ellison et al. [31, 35, 25] are based on the model of owners with delegation. These PKI defining approaches were outlined in Sections 2.2 and 2.4.

An exciting work of Li et al. [61] presents a formal framework (i.e. called RT framework) extending the approach of SPKI/SDSI. This work combines the strengths of RBAC (e.g. separation-of-duty policies) and the strengths of SDSI part of SPKI/SDSI (e.g. threshold structures). Like our approach, the attribute-based access control mechanism discussed in this work allows inferences of attributes and attribute fields. The term attribute corresponds to the term property used in this thesis. The future extensions of our work with respect to integrating distributed RBAC into our approach could profit from the proposal presented in this work.

In Section 10.2.3, we will evaluate these approaches against the security requirements stated in Section 10.2.1.

10.2.2.2. PKI Applying Approaches

Most of the works investigating the application of certificate/credential-based access control treat both PKI models (see Sections 4.4 and 4.3) as competing approaches and base their work on a single PKI model. Even some of these works abstract from any particular PKI model.

Winslet et al. [81, 96] focus on protecting service access at a server and investigate the application of credential-based access control in a Web-based client/server architecture. The authors present the tasks needed for credential management both on the client side and server side. In this work a logic language is used to express control rules. The credential management proposal presented in this work focuses on “credential acceptance policies”. The authors present the requirements of such policies and illustrate a programming methodology for writing such policies by using Prolog. The credentials proposed in this work correspond to accreditation-certificates used in this thesis. The authors state that their approach could be implemented using X.509 certificates, but they do not make any assumption on the underlying PKI model. Subsequent work of Winslet et al. [97, 80] investigates trust negotiation strategies and discusses credential disclosure policies, and presents the privacy vulnerabilities during trust negotiation.

Bonatti et al. [20] introduces a credential-based formal framework and a model to regulate access and information release over the Internet. Besides using credentials, this work also employs data declarations. The data declarations are the statements which are not presented in the credentials. This proposal presents service accessibility rules and portfolio disclosure rules that regulate negotiation between clients and servers. Service accessibility rules are used by servers to specify restrictions with respect to offered services. Portfolio disclosure rules are used by both clients and servers to specify restrictions regulating disclosure of their data declarations and credentials. The authors demonstrate a language with its formal semantics for expressing access and disclosure policies. They also present a policy filtering mechanism used by clients and servers to exchange their requirements in a privacy-preserving manner. This work employs the terms certificate and credential interchangeably and does not make any assumption on the underlying PKI model.

In the Trust Establishment Project at the IBM Hafia Research Laboratory, Herzberg et al. [49, 48] present a system for establishing trust between clients and servers of different organizations according to policies that specify constraints on the contents of certificates. The authors propose an XML-based Trust Policy Language (TPL) which is designed to map the holders of certificates to roles based on the properties encoded in the holders' certificates, a role-assignment policy set by the owner of the service being seeked, and on the roles of the certificate issuers. In [49] servers use a certificate collector to gather missing certificates from issuer sites. These certificates correspond to accreditation-certificates presented in this thesis. However, the syntax of their certificate format is based on the format of X.509 certificates. The subsequent work of Herzberg et al. [48] presents a credential framework for relying applications. The credential framework converts credentials with different formats to a common interface which is to be used by credential relying applications. The authors also propose the use of so-called membership certificates with which the resource owners may delegate the membership decisions to the trusted delegators. However, the authors do not give any details about the infrastructure underlying the delegation. This work also focuses on X.509 certificates and concentrates on unifying different credential formats. The proposals in [49, 48] use the terms role and group interchangeably. These approaches may be useful for cross-organizational access control environments where there is a priori organizational base for mutual trust between the participating organizations. We infer from the given examples that the the authors implicitly assume a common set of properties (e.g. membership in a hospital) shared by the organizations of a client and a server involved in a transaction.

The *safe dealing* approach of Gladney [40], describing an exciting decentralized model, addresses the cross-organizational access control issues of institutional users. This approach is based on service agreements between the organizations of requesting users and resource owners. In the safe dealing approach, there are *enrollments* and *tickets*. An enrollment corresponds to a personal attribute (e.g. group membership). A ticket is a kind of capability aggregating a set of privileges. Enrollments are defined by the service requesting organizations. Tickets are defined by the serving organization. According to this approach, a service agreement defines the mappings of enrollments on the tickets. After a service agreement is executed between these organizations, the service requesting organization issues to each enrolled user a certificate encoding the user's enrollments. Such a certificate comprises the enrollments of a user, the user's identity and her public key. Whenever a user needs access to a protected resource, she sends a request and her enrollment certificates in order to obtain the tickets. The resource owner verifies the user's request on the basis of submitted certificates. In the case of a positive evaluation, the resource owner issues a certificate which encodes the tickets authorized by the corresponding enrollments. The granting of tickets is based on the previously defined mappings between the enrollments and tickets. Like the approach of Herzberg et al. [49, 48], the approach of Gladney may be useful for distributed computing systems where there exists a mutual trust between organizations. The trust relationships can be regulated by contractually involved cross-certification [88, 53], with which is hard to cope, as demonstrated in Section 5.2. A subsequent work of Gladney et al. [41] presents an application of safe dealing approach in the framework of access control management for digital libraries.

In his doctoral thesis, Nikander [66] presents an agent-based security architecture and an object-oriented protocol development framework. This work shows how the proposed security architecture can be based to support distributed trust and policy management using SPKI authorization certificates. For this purpose, a policy manager module has been implemented. Considering the basic components of this module presented in [59], Nikander et al. have followed an approach which is similar to our approach of security module (see Section 9). Compared to our approach, their policy manager is implemented to handle only chains of SPKI authorization certificates, while our security module can handle both SPKI-like delegation chains and X.509-like certificate chains.

Another application based on the approach of SPKI is proposed by Aura et al. [10]. The authors suggest the use of authorization certificates for access control in intelligent networks (IN), where code modules from competitive service providers need to coexist and cooperate on switching platforms.

Johnston et al. [87] discuss the design and implementation of the Akenti access control system. This system enables stakeholders to remotely and securely create and distribute conditions (policy assertions) that must be met by a user desiring access to a resource. The authors employ so-called use-condition certificates. These certificates enable distributed stakeholders to share control over access to resources. In this proposal, a user is represented by her X.509-based identity certificates and attribute certificates which encode a user's attributes (e.g. membership in an organization). A use-condition certificate specifies the combinations of attributes required to access a certain resource. Such a certificate without its signature corresponds to an authorization object in terms of Section 6.4.1 of this thesis. In the architecture of Johnston et al., at time of a resource access, after verification of a user's identity and mutual authentication between the user and the server, the Akenti policy engine gathers use-condition certificates from the stakeholders' servers. The attributes required by use-conditions are verified by obtaining attribute certificates (e.g. associated with the user) from trusted directories. If the use-conditions are met, the policy engine grants access to the requested service. Otherwise, the access is denied.

The following approach of Sandhu et al. [70] is concerned with controlling access to the resource of a single organization. The authors present a certificate-based approach to apply role-based access control (RBAC) on the Web. They employ smart-certificates for access control purposes. A smart-certificate is a typical X.509 identity certificate whose extension fields include X.509 attribute certificates. Each attribute certificate encodes the role information pertaining to a role granted to a user. If the user has roles granted by a single issuer as role server, the role information is encoded in the "organization unit" field of the X.509 identity certificate which is signed by the single issuer. If the user has additional roles granted and certified in a separate attribute certificate issued by another attribute authority, this attribute certificate is put into an extension field of the basic identity certificate. The attribute authority signs on the basic identity certificate together with the attribute added and puts the signature to another extension field in the basic identity certificate. After different attribute authorities repeat this process multiple times, the basic identity certificate becomes a smart-certificate. We argue that this identity-based approach is only applicable in centralized systems or closed distributed systems (e.g. government, military) where the mutual trust between attribute authorities as role servers are high. In other cases, this approach would fail, since a user has to reveal all of her roles encoded in her smart-certificate to an attribute authority which is willing to certify a new role granted to the user by this authority.

10.2.3. Evaluation of Existing Approaches

The basic approach of widespread X.509 public-key infrastructure [4, 51] fails to fulfill the security requirements stated in Section 10.2.1, since X.509 supports certified characterizing properties only in conjunction with the corresponding certified identifying properties. Thus, the pure design of this approach does not enable the use of anonymity techniques.

When we designed our hybrid PKI model, we adopted the observations of the capability-based trust management systems. These systems state that the identities are neither necessary nor useful in open distibuted systems. They also state that the open distributed systems need a delegation-based distributed authorization approach. This approach is significantly different from traditional authorization in centralized systems or in closed distributed systems. The capability-based trust management systems PolicyMaker [17, 18] and KeyNote [16] of Blaze et al., and SPKI of Ellison et al. [33] do not match well the nature of property-based security policies required by open distributed systems like i-mediation and f-mediation. These capability-based trust management systems are designed to delegate asserted permissions, each of which acts as a capability giving the subject certain resource specific permissions. Using the pure designs of these systems, one cannot express the statement that any entity which has an assigned characterizing property is entitled to access some data.

Previous work closest to ours, with respect to authority delegation, is the approach of SPKI/SDSI. The naming scheme [76, 6] proposed in the SDSI part of SPKI/SDSI allows the delegation of property authority to entities having certain properties through the linked names.

In contrast to our approach, SDSI does not explicitly distinguish between free properties and bound properties. Seen from the viewpoint of this thesis, a SDSI local name can be interpreted as a bound authorization attribute within the local name space where the local name is defined. Outside the local name space, such a local name can be interpreted as a free property.

SDSI proposes a solution for distributed computing environments where there exists no global name space or even a hierarchy of name spaces. According to SDSI, an informational environment comprises a collection of local name spaces built bottom-up in a distributed manner. In contrast to SDSI based on such an ideal informational environment, our approach aims to unify and extend the basic PKI models, the model of trusted authorities with licencing and the model of owners with delegation. Thus, in the informational environment underlying our approach, there are local name spaces owned and administrated by resource owners defining bound properties and corresponding administrative properties, and also a hierarchy of trusted authorities defining free properties and corresponding administrative properties. Additionally, in contrast to SDSI, we support the use of conjunction of properties when making inferences about other properties (see Section 8.2.1). Further on, we support the use of fielded properties (e.g. [`name`=age, `value` 18]). We argue that this is a useful feature to infer additional properties based on these field values and to grant a bound property to a certain entity having free properties with certain specific fields.

In the research area of certificate/credential-based access control in open environments, SPKI/SDSI and the proposal presented in [61] are the works that are most promising out of those presented above.

Most of the PKI applying approaches presented above propose access control solutions for either cross-organizational systems based on cross-certification or more open environments where trust evaluation is based on global trusted authorities with licencing or on the model of owners with delegation. All of these PKI applying approaches would fail in open distributed systems where the security domains are incompatible with each other and/or trust between these domains are low or nothing.

We argue that none of these PKI defining and PKI applying approaches can fully fulfill the requirements stated in Section 10.2.1. As seen above, previous approaches for defining and applying a PKI are classified as based either on trusted authorities with licencing or on owners with delegations. In Section 4 of this thesis, we identified the similarities and the differences of these approaches. Roughly summarized, they are similar with respect to the need of supporting documents for a main document, a certificate or a credential, but they differ with respect to how trust evaluations are performed. We argue that many applications require to use and to link both kinds of PKI. Accordingly, we presented a hybrid PKI model which unifies and extends the previous approaches. As demonstrated in Section 5, our solution to certificate/credential-based access control appears to be worthwhile for a broad spectrum of applications.

11. Conclusions and Future Work

In this chapter we conclude this thesis by summarizing the achieved results and identifying some directions for future work.

11.1. Conclusions

The main result of this thesis is a *comprehensive* and *flexible* security framework for mediation between strangers. The achieved framework is comprehensive, since it proposes security solutions for two kinds of mediation which are needed to be employed in open distributed computing systems, in order to master the challenges that arise due to the following facts:

- A client (requesting entity) may aim at identifying promising sources (authorizing entity) which can be quite *heterogeneous* and *autonomous*, not only from the functional point of view, but also with respect to all aspects of security.
- Whatever data a source has to offer, it may aim at supporting a *wide range* of potential clients, which are in general unknown in advance and may belong to *heterogeneous* and *autonomous* security domains.
- Often, there is no relationship between a requesting entity and an authorizing entity prior to a request. Furthermore, these entities belong most probably to different (incompatible) security domains.

Our security framework for mediation is flexible, since the proposed solutions, such as the hybrid PKI model and the agent PKI framework, can be used not only for mediation, but for many other applications.

With the development of the proposed security framework, we made following main contributions:

- We discussed the needs for secure mediation and classified it as *i*-mediation and *f*-mediation according to different employment goals of mediators in distributed information systems. We identified the main characteristics of a security framework supporting property-based security policies which are suitable for secure *i*-mediation and secure *f*-mediation. These issues were discussed in Chapter 3.

- We defined a *flexible* hybrid PKI model which unifies and extends the previous PKI defining approaches (see Chapter 4). We demonstrate the applicability and flexibility of the hybrid model by presenting two practical application scenarios (see Chapter 5).

- We demonstrated an original secure *i*-mediation solution. We first discussed the security requirements (see Chapter 6.2) for *i*-mediation with an emphasis on confidentiality and authenticity, we then proposed a general design (see Chapter 6.3) of secure *i*-mediation that employs accreditation-certificates for the enforcement of confidentiality and authenticity, and finally, we presented the security architecture (see Chapter 6.3.4) for the prototype of our Multimedia Mediator.

- In order to prove the key concepts of this thesis, we have implemented three interrelated demonstrators: a *flexible agent PKI framework* which provides core PKI services (see Chapter 7), an agent oriented and KQML-based *hybrid PKI demonstrator* (see Chapter 8), and a *demonstrator of the security module* for the prototype of the MMM (see Chapter 9).

The proposed security framework is a scientific contribution that has not been published by other authors. However, the presented solutions must not be understood as the ultimate answer to all mediation-related challenges. Depending on the requirements and motivations, many other solutions are possible, which may be more appropriate for some application domains than the framework described in this thesis.

11.2. Future Work

Due to the rather broad spectrum of issues, several of them could not be addressed in this thesis and some restrictive assumptions and simplifications had to be made. There are a lot of issues for further research and development. It should be noted that the following limitations do not undermine the accomplishments of the thesis and it is intended that they will be rectified as future work.

11.2.1. Issues Related to Secure *I*-Mediation

In relation to secure *i*-mediation, we identify following main issues for further research and development:

- For supporting the client with handling her accreditation-certificates, it is desirable to construct a protocol for negotiating the appropriate set of accreditation-certificates to be sent to the source. This includes algorithms for determining minimal sets of personal authorization attributes sufficient for proving eligibility for a query. For this purpose, promising trust negotiation approaches presented in [97, 80] can be adopted.

- We assumed that there is no a priori trust between the sources and the *i*-mediator. It would be interesting to examine the possibilities arising from a scenario where (some of) the sources trust the *i*-mediator, as it might be the case with intraorganizational database federations.

- Furthermore interference of various query optimization techniques – especially caching or materialization, respectively – with the presented access control mechanisms should be investigated. This gets pressing when we also optimize the authorization expressions as mentioned above. In this case we have to balance competing optimization goals.

- In Section 6.3, we pointed out the *i*-mediator's possible ability to infer secrets from unencrypted queries and the sizes of encrypted answers. As mentioned in Section 6.4, our architecture does not yet implement a countermeasure against this possible attack.

- When we implemented the demonstrator, no real external data source was available in the MMM environment. Hence, the demonstrator does not include an implementation of the *answer encryption policy* for *i*-mediation as explained in Section 6.4.3.5. In an advanced version of the MMM's prototype running with real external data source, the proposed answer encryption policy should be implemented.

- At the time of writing this thesis, the functionality of dynamically generating external views (see Section 6.4.3.3) and the functionality of the (Horn clause) rule base (see Sections 6.4.1 and 9.1.1) have been tested in an off-line manner. These functionalities should be integrated into the future versions of the MMM's prototype.

Other open issues are the exploitation of the full scope of ODL, rapid construction of "authorization wrappers", partial encryption of queries, and more sophisticated treatment of encrypted subanswers, e.g., by uniformly using a *privacy homomorphism* [75, 89] for encrypting subanswers. Such a privacy homomorphism allows performing a subset of typical database operations on ciphertexts as if they were executed on plaintexts.

11.2.2. Issues Related to Hybrid PKI Model

There are various topics for future research in relation to the development of the Hybrid PKI Model.

First of all, several details of the hybrid PKI model need further elaboration, including a formal specification and precise translations for embeddings of previous approaches. Based on these foundations, the prototype implementation for the core functionality should be converted into a more mature state. Such an advanced implementation would open to experience various applications besides secure mediation. For such applications, we need guidelines to specify and to enforce security policies in terms of free and bound properties, together with licencing and delegation, in order to exploit the full potentials of the hybrid PKI model. An advanced implementation would also allow to evaluate the actual performance of our approach in terms of effectiveness and efficiency.

Bibliography

[1] ANSI ASC X9 - Public key cryptography for the financial services industry: Certificate management. `http://webstore.ansi.org/ansidocstore/dept.asp`.

[2] The Apache XML project. `http://xml.apache.org/`.

[3] ITU-T recommendation X.509: The directory - authentication framework, 1997.

[4] ITU-T recommendation X.509: The directory - public-key and attribute certificate frameworks, 2000.

[5] IBM XML Security Suite. `http://alphaworks.ibm.com/tech/xmlsecuritysuite/`, Apr. 2002.

[6] M. Abadi. On SDSI's linked local name spaces. *Journal of Computer Security*, 6(1/2):3–21, 1998.

[7] C. Altenschmidt, J. Biskup, U. Flegel, and Y. Karabulut. Secure mediation: Requirements, design and architecture. *Journal of Computer Security*. to appear.

[8] 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, Sept. 1998.

[9] C. Altenschmidt, B. Goz, and F. Peters. The design of a Multimedia Mediator. Technical report, Department of Computer Science, University of Dortmund, Mar. 2000.

[10] T. Aura, P. Koponen, and J. Räsänen. Delegation-based access control for intelligent network services. In *Proceedings of the ECOOP Workshop on Distributed Object Security*, Brussels, Belgium, July 1998.

[11] F. Bancilhon, C. Delobel, and P. Kanellakis, editors. *Building an Object Oriented Database System: The Story of O_2*. Morgan Kaufmann, San Mateo, CA, 1992.

[12] S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. In *Proceedings of the USENIX 1991 Winter Conference*, Dallas, USA, Jan. 1991.

[13] E. Bertino, F. Buccafurri, E. Ferrari, and P. Rullo. An authorization model and its formal semantics. *Journal of Computer Security*, 8(2/3):109–140, 2000.

[14] J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. A mediator for multimedia systems. In *Proceedings of the 3rd International Workshop on Multimedia Information Systems*, pages 145–153, Como, Italy, Sept. 1997.

[15] 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, Nov. 1997. Springer Verlag, 1997.

[16] M. Blaze, J. Feigenbaum, and A. Keromytis. The KeyNote trust management system version 2. RFC 2704, IETF, Sept. 1999.

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

[18] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance checking in the policymaker trust management system. `http://www.research.att.com/library/trs/`, Mar. 1998.

[19] M. Blaze, J. Ioannidis, and A. D. Keromytis. Trust management for IPsec. *ACM Transactions on Information and Security*, 5(2):95–118, May 2002.

[20] P. Bonatti and P. Samarati. Regulating service access and information release on the web. In *Proceedings of the 7th ACM Conference on Computer and Communication Security*, pages 134–143, Athens, Greece, Nov. 2000.

[21] G. Booch, I. Jacobson, and J. Rumbaugh. *The Unified Modeling Language User Guide*. Addison-Wesley, 1998.

[22] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. The MIT Press, 2000.

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

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

[25] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[26] F. Cuppens and A. Gabillon. Rules for designing multilevel object-oriented databases. In J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. 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, Sept. 1998. Springer-Verlag.

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

[28] S. D. C. di Vimercati and P. Samarati. Authorization specification and enforcement in federated database systems. *Journal of Computer Security*, 5(2):155–188, 1997.

[29] D. Eastlake, J. Reagle, and D. Solo. XML signature core syntax and processing. `http://www.ietf.org/rfc/rfc3275.txt`, work in progress, RFC 3275, Internet Engineering Task Force, Mar. 2002.

[30] C. Ellison. SPKI/SDSI and the Web of trust. `http://world.std.com/~cme/html/web.html`, Apr. 2001.

[31] C. Ellison. SPKI/SDSI certificates. `http://world.std.com/~cme/html/spki.html`, Aug. 2001.

[32] C. Ellison. Improvements on conventional PKI wisdom. In *1st Annual PKI Research Workshop*, Gaithersburg, Maryland, USA, Apr. 2002.

[33] C. Ellison, B. Frantz, B. Lampson, R. Rivest, and T. Ylonen. SPKI certificate theory IETF RFC 2693. `http://www.ietf.org/rfc/rfc2693.txt` RFC 2704, Internet Engineering Task Force, Sept. 1999.

[34] C. Ellison and B. Schneier. Ten risks of PKI: what you are not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[35] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. Simple public key certification. Internet draft, work in progress. `http://www.ietf.org/ids.by.wg/spki.html`, June 1999.

[36] D. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information Systems Security*, 4(3):224–274, Aug. 2001.

[37] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO'86*, LNCS 263, pages 186–194. Springer-Berlin, Aug. 1986.

[38] T. Finin, Y. Labrou, and J. 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/`.

[39] B. Gladman, C. Ellison, and N. Bohm. Digital signatures, certificates and electronic commerce. `http://jya.com/bg/digsig.pdf`, April 1999.

[40] H. M. Gladney. Safe deals between strangers. Technical report, IBM Research Report RJ 10155, July 1999. `http://xxx.lanl.gov/ftp/cs/papers/9908/9908012.pdf`.

[41] H. M. Gladney and A. Cantu. Authorization management for digital libraries. *Communications of the ACM*, 44(5):63–65, May 1999.

[42] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Springer-Verlag, 1998.

[43] L. Gong. A secure identity-based capability system. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 56–63, Oakland, California, May 1989.

[44] N. Haller, C. Metz, P. Nesser, and M. Straw. A one-time password system. `http://www.ietf.org/rfc/rfc2289.txt` RFC 2289, Internet Engineering Task Force, Feb. 1998.

[45] M. Harrison, W. Ruzzo, and J. Ullman. Protection in operating systems. *Communications of the ACM*, pages 461–471, Aug. 1976.

[46] Q. He, K. P. Sycara, and T. Finin. Personal Security Agent: KQML-Based PKI. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 377–384. ACM Press, 1998.

[47] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[48] A. Herzberg and Y. Mass. Relying party credentials framework. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference*, LNCS 2020, pages 328–343, San Francisco, CA, 2001.

[49] A. Herzberg, J. Mihaeli, Y. Mass, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: Assigning roles to strangers. In *IEEE Symposium on Security and Privacy*, Oakland, USA, May 2000.

[50] P. Hildebrand. Design and implementation of a KQML-based hybrid PKI for secure mediation. Master's thesis, Department of Computer Science, University of Dortmund, Apr. 2002. german only.

[51] IETF X.509 Working Group. Public-key infrastructure (X.509). `http://www.ietf.org/html.charters/pkix-charter.html`, 1998.

[52] S. Jajodia, P. Samarati, V. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD'97*, pages 474–485, Tucson, AZ, USA, May 1997.

[53] C. Johnson and M. Mancuso. The weakest links. `http://www.intelligententerprise.com/010327/trust1_1.shtml?ebusiness`, Mar. 2001.

[54] 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.

[55] D. Jonscher and K. R. Dittrich. Argos – a configurable access control system for interoperable environments. In *Database Security, IX: Status and Prospects, Proceedings of the 9th Annual IFIP WG 11.3 Working Conference on Database Security*, pages 43–60, Rensselaerville, New York, 1995.

[56] S. T. Kent. Internet privacy enhanced mail. *Communications of the ACM*, 36(8):48–60, Aug. 1993.

[57] C. Kröger. Integration of access control systems into a credential-based security environment. Master's thesis, Department of Computer Science, University of Dortmund, Sept. 2000. german only.

[58] P. Lehmann. Design and implementation of a credential manager. Master's thesis, Department of Computer Science, University of Dortmund, Apr. 2002. german only.

[59] I. Lehti and P. Nikander. Certifying trust. In *First International Workshop on Practice and Theory in Public Key Cryptography*, pages 83–98, San Diego, California, Mar. 1998. IEEE.

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

[61] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, Berkeley, California, USA, May 2002.

[62] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*. to appear.

[63] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, Sept. 1990.

[64] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, Dec. 1978.

[65] B. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9):33–38, Sept. 1994.

[66] P. Nikander. *An architecture for authorization and delegation in distributed Object-Oriented Agent Systems*. PhD thesis, Helsinki University of Technology, Mar. 1999.

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

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

[69] X. Orri and J. M. Mas. SPKI-XML certificate structure. Internet Draft draft-orri-spki-xml-cert-struc-00.txt, work in progress, Internet Engineering Task Force, Nov. 2001.

[70] J. S. Park and R. Sandhu. RBAC on the Web by smart certificates. In *Proceedings of the 4th ACM workshop on role-based access control*, pages 1–9, Fairfax, Virginia, USA, 1999. ACM Press.

[71] E. Parliament and Council. Official Journal of the European Communities L13/12-20, Dec. 1999. `http://www.qlinks.net/comdocs/elsig/en.pdf`.

[72] C. Perkins and P. Calhoun. Mobile IPv4 challenge/response extensions. `http://www.ietf.org/rfc/rfc3012.txt` RFC 3012, Internet Engineering Task Force, Nov. 2000.

[73] 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-09.txt`, May 2000.

[74] J. Reagle. XML signature requirements. `http://www.ietf.org/rfc/`, work in progress, RFC 2807, Internet Engineering Task Force, July 2000.

[75] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In R. DeMillo et al., editors, *Foundations of Secure Computation*, pages 169–177. Academic Press, New York, 1978.

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

[77] 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 (FOSAD 2000)*, LNCS 2171, pages 137–196. Springer, Berlin, 2001, 2000.

[78] P. Samarati, M. Reiter, and S. Jajodia. An authorization model for a public key management service. *ACM Transactions on Information and System Security (TISSEC)*, 4(4):453–482, Nov. 2001.

[79] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 2:38–47, 1996.

[80] K. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, and J. Jacobsen. Protecting privacy during on-line trust negotiation. In *2nd Workshop on Privacy Enhancing Technologies*, San Francisco, CA, Apr. 2002. to appear.

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

[82] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, Sept. 1990.

[83] Sun Microsystems. Java 2 SDK documentation. `http://java.sun.com/j2se/1.3/docs.html`.

[84] Sun Microsystems. Keytool - Key and Certificate Management Tool. `http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html`.

[85] 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 of the 10th IFIP WG 11.3 Working Conference on Database Security*, pages 23–42, Como, Italy, 1997. Chapman & Hall.

[86] C. Thirunavukkarasu, T. Finin, and J. Mayfield. Secret agents - a security architecture for the KQML agent communication language. In *4th International Conference on Information and Knowledge Management - Workshop on Intelligent Information Agents*, Baltimore, Maryland, USA, Dec. 1995.

[87] W. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the 8th USENIX Security Symposium*, Washington D.C., Aug. 1999.

[88] J. Tumbull. Cross-certification and PKI policy networking. `http://www.entrust.com/resources/pdf/cross_certification.pdf`, Aug. 2000.

[89] N. R. Wagner, P. S. Putter, and M. R. Cain. Encrypted database design: specialized approaches. In *IEEE Symposium on Security and Privacy*, pages 148–153, 1986.

[90] S. Weeks. Understanding trust management systems. In *IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 2001.

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

[92] G. Wiederhold, M. Bilello, and C. Donahue. Web implementation of a security mediator for medical databases. In T. Y. Lin and S. 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.

[93] G. Wiederhold, M. Bilello, V. Sarathy, and S. Qian. A security mediator for health care information. In *Proceedings of the 1996 AMIA Conference*, pages 120–124, Washington DC, Oct. 1996. Journal of the AMIA.

[94] G. Wiederhold, M. Bilello, V. Sarathy, and X. Qian. Protecting collaboration. In *Proceedings of the National Information Systems Security Conference (NISSC'96)*, pages 561–569, Baltimore MD, Oct. 1996.

[95] G. Wiederhold and M. Genesereth. The conceptual basis for mediation. *IEEE Expert, Intelligent Systems and their Applications*, 12(5):38–47, Sept.-Oct. 1997.

[96] M. Winslett, N. Ching, V. Jones, and I. Slepchin. Assuring security and privacy for digital library transactions on the web: Client and server security policies. In *Proceedings of Forum on Research and Technical Advances in Digital Libraries (ADL'97)*, pages 140–152, Washington, DC, May 1997.

[97] T. Yu, M. Winslett, and K. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of 8th ACM Computer Conference on Computer and Communication Security*, pages 146–155, Philedelphia, Pennsylvania, Nov. 2001.

[98] P. R. Zimmermann. *The official PGP User's Guide*. MIT Press, 1996.

Curriculum Vitae

Last name:	Karabulut
First name:	Yücel
Day of birth:	1 March 1970
Birthplace:	Sivas, Turkey

1975 - 1983	Secondary School, Sivas
1983 - 1986	Gymnasium, Istanbul
1986 - 1991	Ege University, Izmir
	studies in computer engineering; graduation in 1991
1992 - 1996	University of Dortmund
	studies in computer science; graduation in 1996
since 1996	Research and Teaching Assistant at the Information Systems and Security Group
Sept. 2002	PhD Thesis: “Secure Mediation Between Strangers in Cyberspace”