

Secure mediation: requirements, design, and architecture

C. Altenschmidt, J. Biskup, U. Flegel and Y. Karabulut *

Universität Dortmund, Informatik VI, D-44221 Dortmund, Germany

Tel.: +49 231 7552641; Fax: +49 231 7552405;

E-mail: {altensch,biskup,flegel,karabulu}@ls6.cs.uni-dortmund.de

In mediated information systems clients and various autonomous sources are brought together by mediators. The mediation paradigm needs powerful and expressive security mechanisms considering the dynamics and conflicting interests of the mediation participants. Firstly, we discuss the security requirements for 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. Using a public key infrastructure such personal authorization attributes can be bound to asymmetric encryption keys by credentials. Secondly, we propose a general design of secure mediation where credentials are roughly used as follows: clients show their eligibility for receiving requested information by the contained personal authorization attributes, and sources and the mediator guarantee confidentiality by using the contained encryption keys. Thirdly, we refine the general design for a specific approach to mediation, given by our prototype of a Multimedia Mediator, MMM. Among other contributions, we define the authorization model and the specification of query access authorizations within the framework of ODL, as well as the authorization and encryption policies for mediation, and we outline the resulting security architecture of the MMM. We also analyze the achievable security properties including support for anonymity, and we discuss the inevitable tradeoffs between security and mediation functionality.

1. Introduction

Recent trends in information technologies led to vastly improved communication facilities like the Internet, an explosion of on-line information providers, and challenging new demands of users. Whenever a user looks for a piece of information, she may aim at identifying promising sources, which can be quite heterogeneous and autonomous, and then retrieving and integrating the required data. And whatever data a source has to offer, it may aim at supporting a wide range of potential clients, which in general are unknown in advance. According to these trends, various forms of interoperable information systems have been developed. While federated database systems [26,33,40] have already come into existence, increasingly ambitious further demands have evolved and resulted in the paradigm of *mediated information systems* [46,48]. Some current projects on mediation are TSIMMIS [44], HERMES [41], Information Manifold [32], SIMS [3], AURORA [49], DISCO [43], Squirrel [27], DIOM [34], Garlic [16], OBSERVER [35], InfoSleuth [5], and MMM [9,10].

*Corresponding author.

In mediated information systems a client, seeking for 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 the heterogeneity and the autonomy of the sources, not only from the functional point of view but also with respect to all aspects of *security*. This includes confidentiality and authenticity, as well as integrity, anonymity, accountability and availability. As the first contribution of this paper, we discuss these security requirements for 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, 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. This includes, as a special case, attributes identifying the user, but in many situations identification of a user is neither needed nor appropriate for 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 interpret submitted personal authorization attributes, i.e., which attributes they require in order to return some information.
- 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 can take advantage of the evolving *public key infrastructure*. Basically, two services are exploited: credentials and asymmetric encryption. The concept of *credentials* has been advocated by Chaum [18] for supporting privacy in networked systems. Since then it has been adopted for various purposes in interoperable systems, for electronic payment and marketplaces as well as for middleware systems. Further credential proposals and standardization efforts include [11,14,22,23].

Now credentials potentially provide for an inherently scalable and secure mechanism for managing certified personal authorization attributes in an open environment, as required for mediation. Additionally, credentials allow to securely bind public encryption keys to personal authorization attributes. Thus we are led to the second contribution of this paper, a general design of secure mediation. Shortly summarized, secure 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 *credentials*.
- A mediator examines, selects and forwards submitted credentials together with appropriate subqueries to the data sources.

- A data source autonomously bases *access decisions* for requested data on shown credentials, and it returns subquery answers in *encrypted form* to the mediator. For encryption a data source applies the user's *public key* which is also contained in the credentials for an asymmetric encryption scheme.
- Then the 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 mediation apply to a wide range of mediation technologies. In order to implement the general design, the specific approach to mediation has to be taken into account. As the third contribution of this paper, we present the *security architecture* for the prototype of our Multimedia Mediator [9,10].

The *Multimedia Mediator*, 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 a 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).

A *security module* for the MMM is being constructed under the following requirements: (1) implement the design as outlined above, (2) smoothly integrate the security features into the functionalities of the MMM prototype, and (3) meet emerging proposals and standardization efforts [14,22,23] for credentials.

The resulting architecture of this security module exhibits the following features:

- an *authorization model* that allows considering expressions over *personal authorization attributes* extracted from *credentials* as grantees, to be used for representing the query access authorizations of the sources;
- specifications of query access authorizations as *annotated schema declarations*, which can be communicated from the sources to the MMM, in particular;
- a *schema authorization policy for mediation*, which allows dynamically generating external views for spontaneous users;
- an *instance authorization policy for mediation*, which conjunctively combines the access restrictions of the mediator and the sources;
- an *answer encryption policy for mediation*, which supports information flow control;
- an *isolated co-existence* of the functional layers and the security layers treating authorization data in close correspondence to functional data;
- user support for *credential management* by determining implications among sets of personal authorization attributes.

The security requirements on mediation, the general design of secure mediation, and the specific MMM security architecture have been introduced in previous reports [2,7,8]. This paper provides an integrated and extended view on these issues and thereby summarizes the achievements of our project. The rest of the paper is structured as follows. In Section 2, Requirements, we discuss the needs of secure mediation, thereby emphasizing the differences and the new challenges in comparison to the more traditional federated approach. In Section 3, Design, we present the general design for secure mediation, and we examine its security properties. In Section 4, Architecture, we specify the details of the prototype implementation for our specific approach to mediation. Finally, Section 5 compares our achievements with related work, and Section 6 addresses conclusions and open issues.

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 mediators clients can find and correlate information more efficiently. Information sources are likely to meet the client's requirements and to cooperate with mediators. Like in a marketplace of supply and demand there exist different motivations for cooperation between the participants in mediation. 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.

While information sources may have cooperation contracts with mediators, it can be assumed that spontaneous clients are unknown beforehand. Clients thus cannot be registered with 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 mediated systems. 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 mediated systems due to their dynamics.

Table 1 summarizes some of those key differences between federated and mediated approaches, which also affect security considerations. These differences as well as the similarities of the two approaches are visualized by Fig. 1 and Fig. 2 and further explained as follows.

Table 1
Federated vs. mediated approach

	federated approach	mediated 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	identity-based	attribute-based
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 and dynamic

In contrast to 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 probably has had significant impact on specific security architectures as often found in federated database systems.

Mediated systems are more suitable to model dynamics and low trust between interacting parties. A mediator's top-down design paradigm (see Fig. 2) allows for a stable data description of integrated information at varying degrees of source fluctuation, whereas a bottom-up approach (see Fig. 1) requires a redesign of the global integration schema each time a local schema changes or is added. 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.

Due to this spontaneity, the employment of merely identity-based authentication and authorization approaches, as traditionally used in federated systems, is less useful for mediated systems. 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 for deciding 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 reveal their identities for private reasons and thus prefer to remain anonymous. Additionally for a resource owner, it may be necessary rather to see evidences of a client's eligibility (e.g., profession, organizational membership, security clearance, academic title) than to know *who* they are.

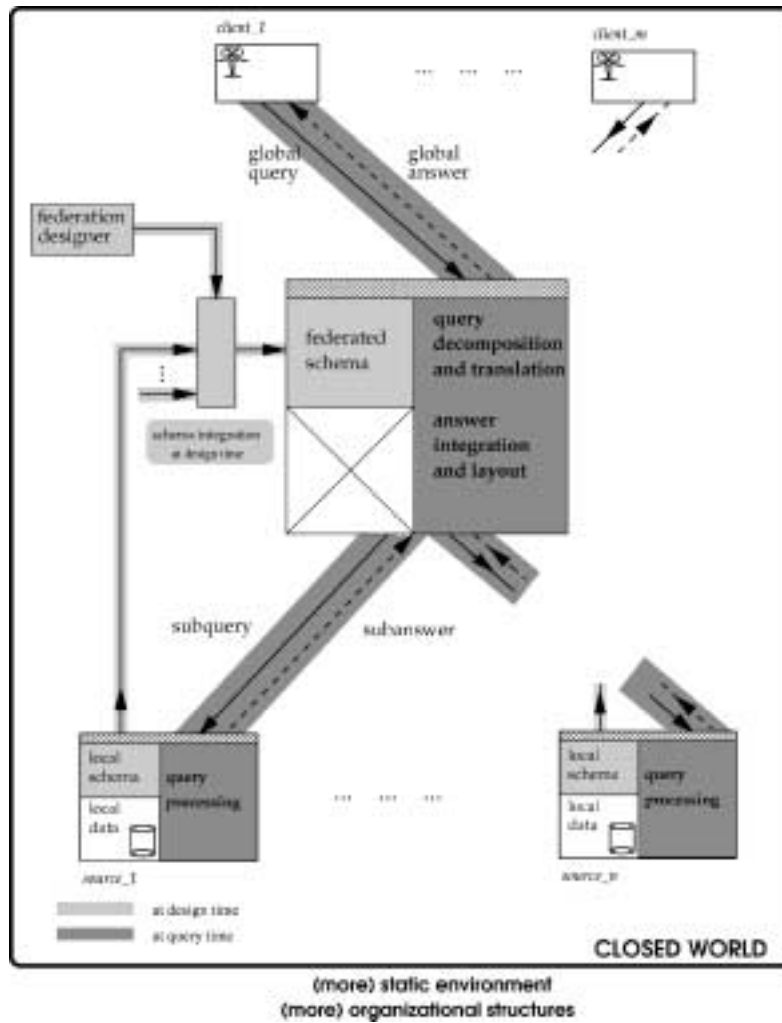


Fig. 1. Design of federated systems.

We consider the following example scenario motivating our secure 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 films and committing crimes of violence. For this purpose, she may need to send at least the following query and evidences of her eligibility to a mediator specialized in forensic and movie sources.

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

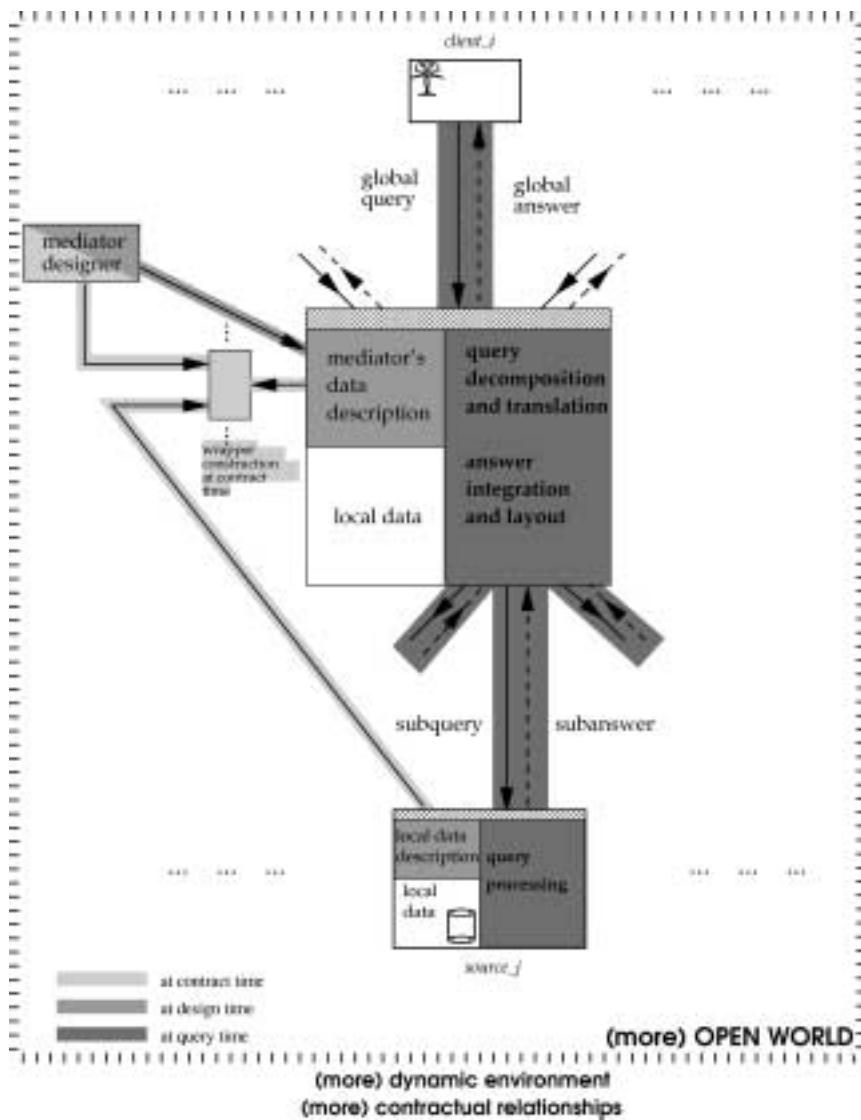


Fig. 2. Design of mediated systems.

In order to answer this query the 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.

3. Design

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 to appropriate clients only. 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 that a client directly addresses a source, as well as when both the client's request and the source's delivery are mediated.

3.2. Basic informational environment

Although the identity-based public-key infrastructures (PKI) are considered as the state of the art for authentication systems, we believe that the sole use of identity-based PKI systems in conjunction with ACLs will be still inadequate for secure mediation. Emerging Internet services like information mediation require powerful and expressive authorization mechanisms taking the privacy and anonymity needs of the clients into account. Key-oriented public-key infrastructures [14,22,23,25] supporting identity-free access control are promising approaches to fulfill the requirements of secure mediation as described above. By using these key-oriented public-key infrastructures and thus by avoiding the use of identities, an important and necessary step forward in achieving anonymity is done.

We assume that there are trusted third parties (TTPs), trusted by all participants of a transaction, that offer at least the following services:

- A TTP signs an *identity certificate* of the rough form $\langle \text{identity}(\text{address}), \text{public encryption key}, \text{public verification key} \rangle$, thereby assuring that the *participant* specified by the first component is the legitimate owner of the matching secret keys for decryption and signing.
- A TTP signs a *credential* of the rough form $\langle \text{attribute}, \text{public encryption key}, \text{public verification key} \rangle$, thereby assuring that the *attribute* specified by the first component is enjoyed by the legitimate owner of the matching secret keys for decryption and signing.

In a secure mediated system the attributes appear in the role of *personal authorization attributes*. That is, our basic protocols employ the attributes as evidence whether the owner of the matching secret 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 got knowledge of the credentials, whether directly from the owner of the matching secret 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 secret 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.

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 credentials 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 credentials 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 credentials, 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. 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 credentials⟩ to the source.

Delivery phase

2. The source verifies each credential 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 occurring in the set of credentials. 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 3.1:

- Clients and sources exclusively have to trust the TTPs that signed the credentials.
- 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.
- Eligibility is supposed to be definable in terms of personal authorization attributes.
- Since personal authorization attributes are shown in the form of credentials 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 from the query as well as the answer.
- Since a source cannot always effectively decide whether two given credentials 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 credentials 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 credentials are definitely not linked.
- Even in an untrusted network, only the unique owner of the verified credentials 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 secret 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 non-deterministic 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 credentials 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 credentials. 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 somebody else's credentials, the correct owner could be erroneously or maliciously blamed for the request. However, in a dispute about the sender of the request nobody 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 credentials, availability of the corresponding public verification keys is required.

3.4. *The design of secure mediation*

We now extend the approach presented in Section 3.3 for the case of 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 a mediator M .
- (b) The 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 mediator 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 mediator M .
- (f) The mediator M integrates the received subanswers d_S into a global answer d .
- (g) The mediator 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 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 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 mediator and the sources, as illustrated in Fig. 3:

- In step (a) the client includes her credentials into the request for query q .
- In step (c), for each of the relevant sources, the mediator selects a subset of credentials which is necessary to answer the subquery q_S . Then these “local credentials” are attached to q_S and sent to source S . As a simple default, the mediator just forwards the received set of credentials.
- 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 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 mediator, acquires knowledge about the client’s credentials and thus could give raise to false blames about the sender of a request. The mediator could also redistribute credentials of clients to third parties or misuse the client’s credentials to validate presumptions about the source’s data. But these possibilities do not introduce substantially new problems.

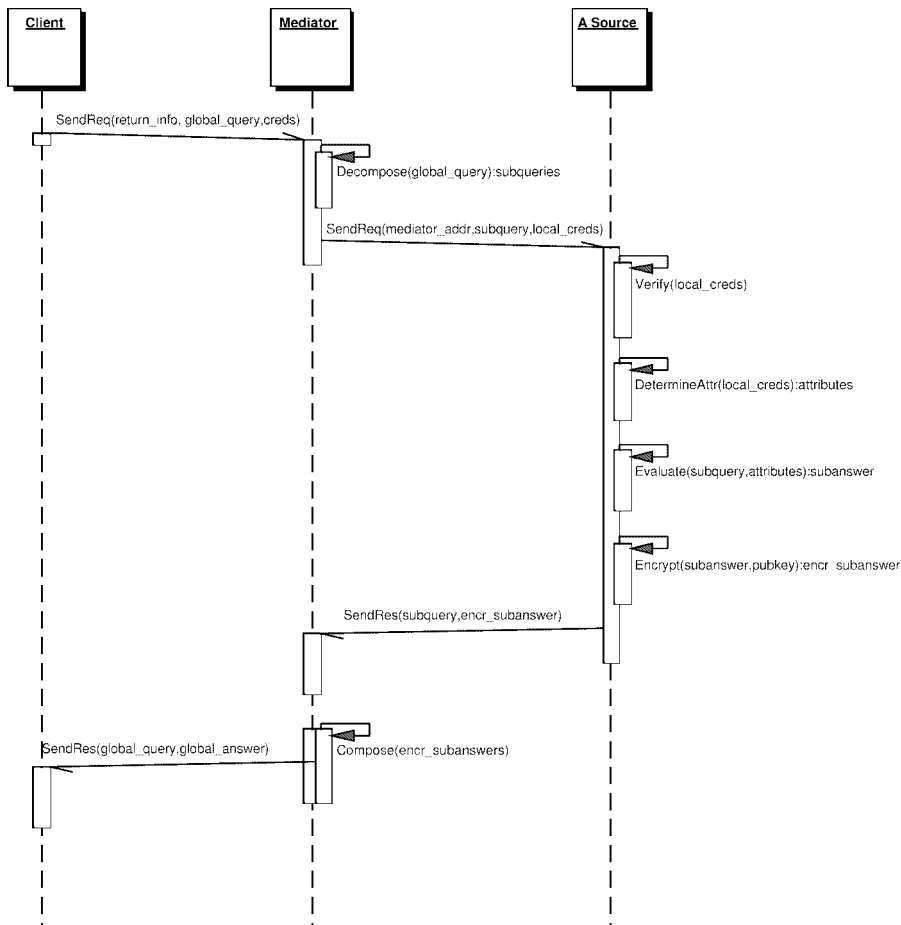


Fig. 3. Protocol for secure mediated query answering; for a legend see [13].

- A mediator may compromise the integrity of queries and answers, if no additional actions are taken. In particular, clients have to trust the mediator for the delivery of complete answers in the context of best effort mediation.
- Since mediators break a client’s global query down to several subqueries for several sources, signatures protecting the entire global query will not be seen and verifiable by the sources. Also the 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 publication.
- Since 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 publication.

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.

3.4.1. *Layered mediation*

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

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 mediator has to integrate the subanswers sent back to the mediator by the sources.
- The mediator should not be able to break the security policies of the sources. In particular, the 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 mediator operates on the ciphertext only. If a datum is not protected, the mediator may operate on the unencrypted datum. In each case no trust in the mediator is necessary.

3.5. *Summarized security properties*

The protocols for secure query answering and for secure mediated query answering as proposed in Sections 3.3 and 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 re-

quest 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 3.3 providing client accountability should not be used.

Use of credentials avoiding the identities of the clients at first glance seems to support anonymity. Eventually, the public keys being included in a credential 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 a credential by linking observations of its included public keys may still be able to build a dossier of an individual. Also mediation raises some additional concerns regarding the privacy of the clients. Mediation requires the clients to pass their requests through mediators. This means that the mediators can compile client profiles and misuse them. Once a client hands over her credential to the mediator, she cannot be sure the mediator retains it for no other purposes except for processing her query. Moreover, mediators will be amiable targets, for a mediator is a kind of channel where the credentials of individuals conglomerate.

We observe that the privacy of the clients cannot be protected by using the credentials if the client does not make proper use of public key technology. This requires the client's awareness of the implications when using credentials. Using the same credential in many requests could be almost as privacy violating as using an identity certificate. Instead, a client can employ multiple credentials specifying the same personal authorization attribute(s) in each request.

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

All security properties of the protocols leverage off the authenticity and integrity of credentials and certificates, respectively. The clients and source therefore need to place trust in the basic informational environment described in Section 3.2. The clients also have to trust the sources to provide answers to their best knowledge, and they need to trust the 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.

4. Architecture

The *Multimedia Mediator*, MMM, is implemented as an autonomously operating agent. It is based on the object-oriented database management system O2 [4]. Figure 4 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 [36] for data exchanges, KQML [24] for agent communications, and ODL/OQL [17] for schema declarations and query expressions.

This architecture supports the design for secure mediation presented in Section 3 in a straightforward way as follows. A client can send a *request*, i.e., a query accompanied by credentials, to the 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 credentials, 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 mediator, which in turn integrates the subanswers and sends the resulting global answer back to the client.

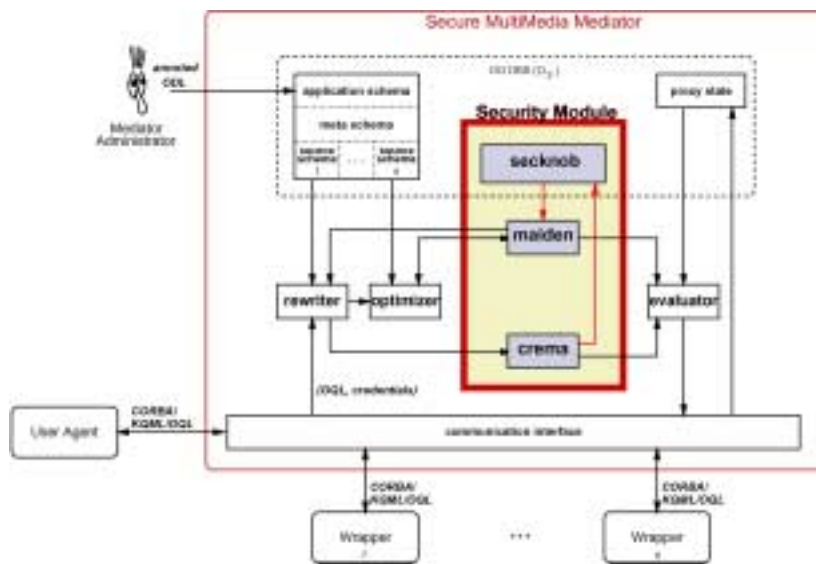


Fig. 4. Security architecture of the Multimedia Mediator.

4.1. Authorization model

The following authorization model explains what we mean by “a client is authorized to access some data”. 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 credentials included in the client’s request. The conjunction r corresponds to identified users or roles in more traditional approaches. For deciding whether r represents a client who is allowed to exercise the privilege p , the 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 5 shows a coarse model for the authorization database. As usual, an *authorization* specifies a grantor, a privilege and a grantee. A privilege is composed of an 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 credential 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*. Note that a client’s representation r , being a conjunction of the personal authorization attributes built from the client’s credentials, 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 ”.

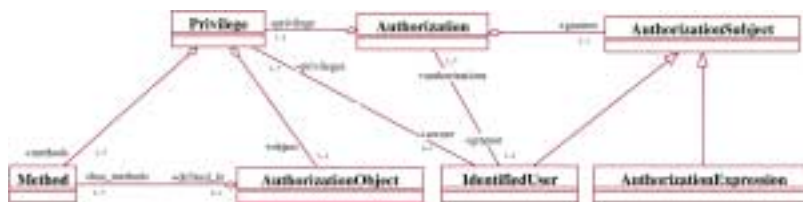


Fig. 5. Coarse model for the authorization database.

In the full model, the exact definition of the relationship “*qualifies for*” formalizes the rough intuition of “*logically implies* under additional assumptions”. 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*.

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 mediator itself applies two more elaborated *authorization policies for mediation*, one for querying schemas and another one for querying instances, which combine the involved qualifications with additional considerations, to be presented in Section 4.3.

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*¹; *attribute type* built from classes and atomic types (*boolean*, *string*, *long*, ...) using constructors like *set*, *list*, *struct*; *relationship* (specifying inverse attributes); *key*; *extent* for denoting an access path to a full class extension. Further advanced features (*method*, *inheritance*, *interface*, *exception*, ...) could be added to future versions of the prototype.

For specifying access authorizations in the context of the MMM, we have to identify the possibly relevant instances (p, g, a) , with privilege $p = (o, m)$ for accessing an authorization object o by method m , grantor g , and authorization expression a . For the current prototype, as well as for the sake of short presentation, we make the following simplifying assumptions:

1. We restrict to *schema based* authorizations (refraining from content based authorizations which depend on actual data values). Thus as authorization objects we allow concrete incarnations of all ODL features.
2. We assume some generic access method for all authorization objects (refraining from specializing the generic method into *read*, *navigate*, and so on).

¹The 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.

Giving these assumptions, we can succinctly express access authorization by adding two kinds of annotations to ODL schema declarations, each of which consists of an authorization expression:

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

4.3. MMM-specific secure mediation

For secure 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 visualizes some of these concepts.

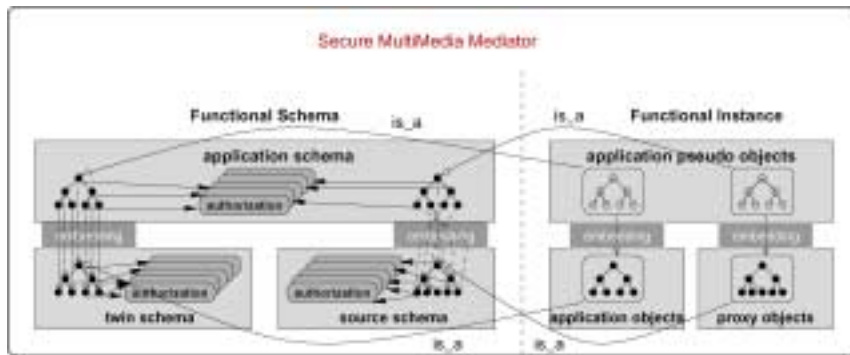


Fig. 6. Schema and instance with authorizations.

Throughout this section we use a running example based on the motivating example scenario exposing the psychoanalyst's query outlined in Section 2. This example will help to understand the tasks handled in each phase.

4.3.1. Initialization phase

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

But for some parts of the schema, the administrator may want to maintain data owned by the MMM itself. Technically, for such parts the administrator additionally declares a *twin schema* as a subschema of the application schema, and 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 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.

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 credentials. The basic functional requirements are that appropriate parts of the application schema can be *embedded* in matching parts of the (virtual or existing) source schema. If the requirements are met, an appropriate *wrapper* agent is created which is devoted to convert source items into a form that is canonical for the mediator, and vice versa.

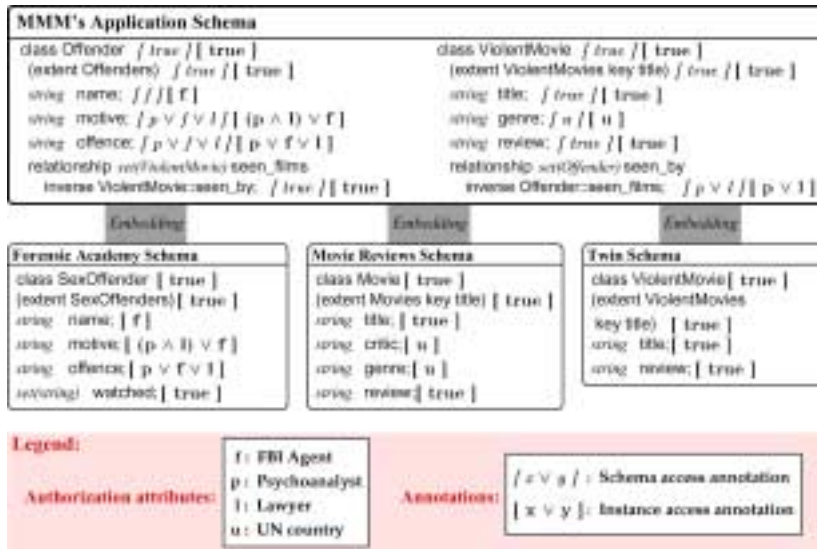


Fig. 7. Example for the initialization phase for the MMM and preparatory phases for the sources.

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 [1]. 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.

Figure 7 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 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 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 Fig. 7, the administrator of the 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.

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 credentials. Let us assume that these credentials 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 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 [19,37]).

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

On demand, the 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 credentials 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 mediation* (see Section 4.3.5). In order to do so we have to interpret the schema access annotations as fictitious instance access annotations on the meta schema. Thus the client gets a protected subschema.

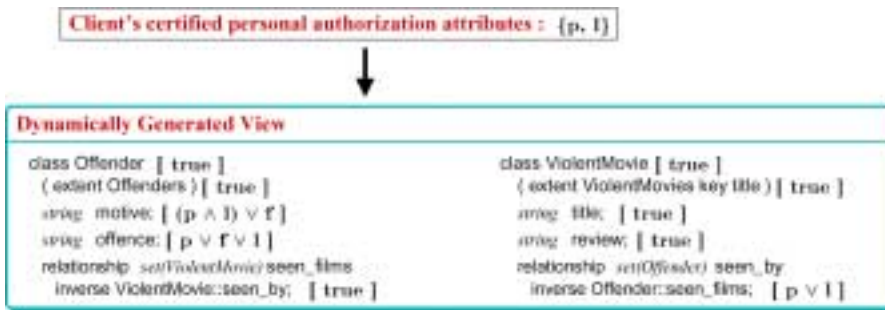


Fig. 8. Example for the preparatory phase for a client.

Figure 8 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`.

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 credentials. The MMM analyzes the request with respect to functional and authorization requirements, thereby identifying subrequests to be forwarded to connected sources. Hereby, the twin source maintained by the MMM itself is treated like any external source, except that no wrapper is involved.

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

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

- A request is (globally) *permitted* (for an authorization expression *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 credentials. The *expectation* is based on the *pure* authorization policy applied to the authorization database generated from the annotations specified for the representation of the source schema.

- If a request is (globally) permitted, then all subrequests are forwarded 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 mediator and forwarded to the client without any further restrictions.

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

Figure 9 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.

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 4.1, and encrypts the subanswer accordingly. More specifically, the source is supposed to proceed according to the following *answer encryption policy for mediation*:

- Within the answer each occurrence 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 credentials.
- 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 the occurrence 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 occurrence of v . Each of these schema items carries an authorization expression as an instance access annotation, the conjunction of which

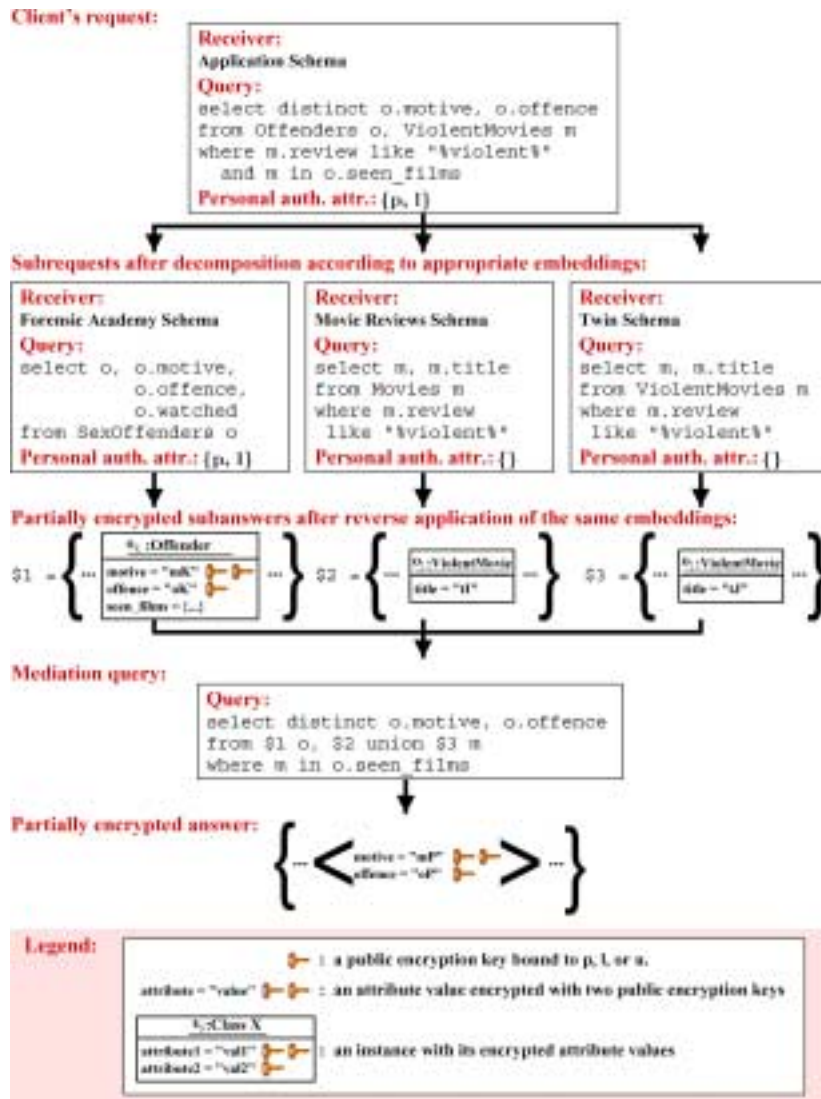


Fig. 9. Example for request and delivery phases.

being 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 occurrence 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 credentials.

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

In our example, shown in Fig. 9, 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 `ViolentMovie` object references. The titles of the referenced `ViolentMovie` objects are the 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 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 mediator.
- We shift the final integration of subanswers to the client who can decrypt the subanswers.

These topics are beyond the scope of the present paper, but we want to roughly illustrate the first strategy. See the modification of our example shown in Fig. 10.

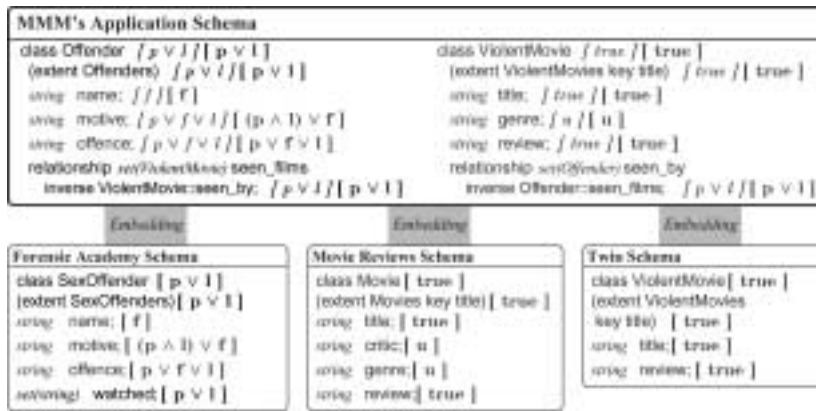


Fig. 10. An alternative scenario with more restricted security policies.

The class SexOffender and its attribute watched are annotated by the authorization expression $p \vee l$ (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 11 depicts a possible solution for this problem. The result of the first stage subrequests is pushed to the forensic academy for the second stage subrequest, which immediately produces the result of the global query.

4.4. The security module

In this section we provide some more details on how *secure mediation* as explained in Section 4.3 is actually implemented as part of the MMM. Section 4.4.1 describes the main security components. And Section 4.4.2 sketches a typical control flow among the functional and security components for a request phase (see Section 4.3 and Fig. 4). This sketch is intended to demonstrate the subtle interactions between functional and security considerations found for mediation.

4.4.1. Main components

The security module of the MMM has three main components, the security knowledge base SECKNOB, the credential manager CREMA, and the mediator authorization decision engine MAIDEN (see Fig. 4).

The security knowledge base SECKNOB maintains the following parts:

- The *authorization databases* (see Section 4.1) that are generated from the annotated declarations of the application schema, the twin schema and the representations of the external source schemas (see Sections 4.3.1 and 4.3.2).
- The *credentials* (see Section 3.2) submitted by a client.

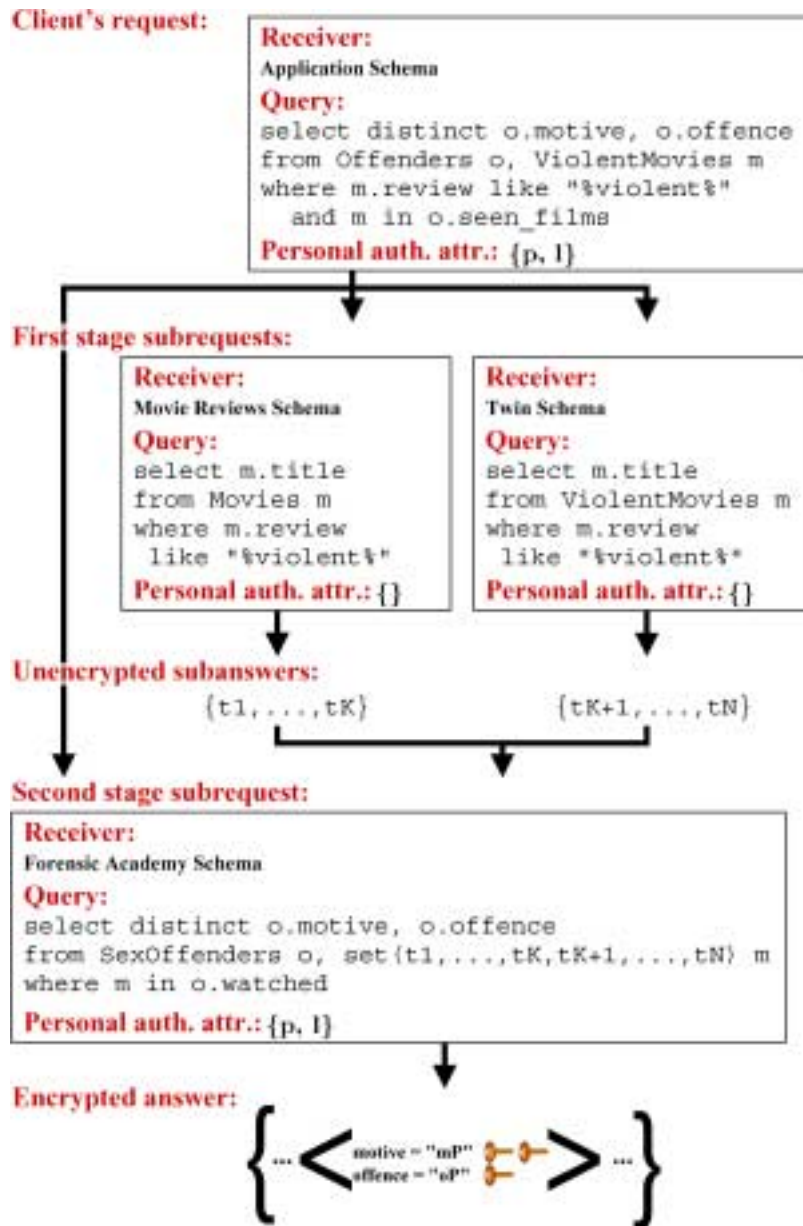


Fig. 11. Query evaluation for the alternative scenario.

- The *personal authorization attributes* (see Section 4.1) extracted from the credentials together with links associating them with the credentials they are extracted from.
- 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 4.1) must be decided. The rule base is intended to take advantage of “real world” properties of personal authorization attributes (see Section 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 [6,29].
- A collection of *services* which unify the various formats proposed in existing and emerging standards for credentials [14,22,23,28].

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

The mediator authorization decision engine MAIDEN implements the *authorization policies for mediation*, both for *schemas* and for *instances* (see Sections 4.3.3 and 4.3.4), and the underlying *pure* authorization policy (see Section 4.1) together with its basic relationship “*qualifies for*” among authorization expressions.

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

4.4.2. Query evaluation with credentials

In this section 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 exceptions are executed):

1. The query rewriter sends the included credentials to CREMA.
2. CREMA verifies the credentials, extracts the personal authorization attributes and inserts the credentials, 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 mediation*.

7. MAIDEN confirms that all requirements of the instance authorization policy for mediation are met, and then it forwards the optimal query plan to the query evaluator.
8. Finally the query evaluator gathers the required credentials with the help of CREMA, produces the subrequests to the sources and forwards them to the twin source and the wrappers, respectively.

5. Related work

Security in interoperable information systems has mostly been 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. [30,31] describe 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. [42] addresses 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. [21] addresses specification and bottom-up derivation of authorizations and consideration of administrative privileges.

For contributions to security in mediated systems see [15,20,29,47]. These works employ identity-based authentication and authorization approaches which appear to be less useful for mediated systems, since the dynamics and conflicting interests of the participants of the mediated systems require an attribute-based authorization approach as discussed in this paper. [29] presents a flexible authorization manager (FAM) that can enforce multiple access control policies within a unified framework. The concepts described there provide the application developer with a set of predefined policy options. A security administrator, using the FAM architecture, may specify an authorization for any object and any client, and may then specify an access control policy. The client may select such policies from an authorization library or extend the library with his own application specific policies to suit his own needs. [47] presents a web implementation of a security mediator regulating access to medical databases. This work involves access control and sanitization of answers before they are returned to the subject. [20] presents a mediator-based solution to the problem of providing secure interoperation among independent information sources protected by mandatory policy restrictions. [15] provides 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 concept of credentials has also been adopted previously for various purposes in interoperable systems, for instance in [12,39].

6. Open issues

The security architecture of the MMM indicates how query access control based on credentials can be implemented for mediation. The ongoing implementation is based on several original contributions, in particular the identification of suitable schema authorization and instance authorization policies for mediation, and ODL declarations with schema access annotations and instance access annotations.

There are a lot of issues for further research and development. For supporting the client with handling her credentials it is desirable to construct a protocol for negotiating the appropriate set of credentials to be sent to the source. This includes algorithms for determining minimal sets of personal authorization attributes sufficient for proving eligibility for a query.

Throughout the paper we assumed that there is no a priori trust between the sources and the mediator. It would be interesting to examine the possibilities arising from a scenario where (some of) the sources trust the 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 3 we pointed out the mediator's possible ability to infer secrets from unencrypted queries and the sizes of encrypted answers. As mentioned in Section 4, our architecture does not yet implement a countermeasure against this possible attack.

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* [38,45] for encrypting subanswers. Such a privacy homomorphism allows performing a subset of typical database operations on ciphertexts as if they were executed on plaintexts.

References

- [1] 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*, Istanbul, Turkey, 1998, pp. 121–129.

- [2] C. Altenschmidt, J. Biskup and Y. Karabulut, Security architecture of the Multimedia Mediator, in: *Data and Application Security: Developments and Directions, Proceedings of the 14th Annual IFIP WG 11.3 Working Conference on Database Security*, Schoorl, Holland, 2000, Kluwer Academic Press, 2001, pp. 77–87.
- [3] Y. Arens, C.A. Knoblock and W. Shen, Query reformulation for dynamic information integration, *Journal of Intelligent Information Systems* **6**(2/3) (1996).
- [4] F. Bancilhon, C. Delobel and P. Kanellakis, eds, *Building an Object Oriented Database System: The Story of O₂*, Morgan Kaufmann, San Mateo, CA, 1992.
- [5] R.J. Bayardo et al., InfoSleuth: Agent-based semantic integration of information in open and dynamic environments, in: *SIGMOD '97*, Tucson, AZ, USA, 1997, pp. 195–206.
- [6] E. Bertino, F. Buccafurri, E. Ferrari and P. Rullo, An authorization model and its formal semantics, *Journal of Computer Security* **8**(2/3) (2000), 109–140.
- [7] J. Biskup, U. Flegel and Y. Karabulut, Secure mediation: requirements and design, in: *Database Security, XII: Status and Prospects, Proceedings of the 12th Annual IFIP WG 11.3 Working Conference on Database Security*, Chalkidiki, Greece, 1998, Kluwer Academic Press, 1999, pp. 127–140.
- [8] J. Biskup, U. Flegel and Y. Karabulut, Towards secure mediation, in: *1st Workshop on Sicherheit und Electronic Commerce*, Essen, Germany, 1998, Vieweg-Verlag, 1999, pp. 93–106.
- [9] 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*, Como, Italy, 1997, pp. 145–153.
- [10] 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*, Brisbane, Australia, Springer-Verlag, 1997, pp. 31–43.
- [11] M. Blaze, J. Feigenbaum and A. Keromytis, The KeyNote trust management system version 2, RFC 2704, IETF, Sept. 1999.
- [12] G. Bleumer and M. Schunter, Privacy oriented clearing for the german health-care system, in: *Personal Information Security, Engineering and Ethics*, R. Anderson, ed., Springer-Verlag, 1997, pp. 175–194.
- [13] G. Booch, I. Jacobson and J. Rumbaugh, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.
- [14] S.A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*, The MIT Press, 2000.
- [15] K.S. Candan, S. Jajodia and V.S. Subrahmanian, Secure mediated databases, in: *12th International Conference on Data Engineering*, S.Y.W. Su, ed., New Orleans, Louisiana, USA, IEEE, IEEE Computer Society Press, 1996, pp. 28–37.
- [16] M.J. Carey et al., Towards heterogeneous multimedia information systems: The Garlic approach, in: *Proceedings of the Fifth International Workshop on Research Issues in Data Engineering (RIDE): Distributed Object Management*, Los Angeles, California, 1995, pp. 123–130.
- [17] R.G.G. Cattell and D. Barry, eds, *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann, San Francisco, 2000.
- [18] D. Chaum, Security without identification: Transaction systems to make big brother obsolete, *Communications of the ACM* **28**(10) (1985), 1030–1044.
- [19] F. Cuppens and A. Gabillon, Rules for designing multilevel object-oriented databases, in: *Proceedings of the 5th European Symposium on Research in Computer Security (ESORICS '98)*, number 1485 in LNCS, J.-J. Quisquater, Y. Deswarte, C. Meadows and D. Gollmann, eds, Louvain-la-Neuve, Belgium, 1998, Springer-Verlag, pp. 159–174.
- [20] S. Dawson, S. Qian and P. Samarati, Providing security and interoperation of heterogeneous systems, *Distributed and Parallel Databases* **8**(1) (2000), 119–145.

- [21] S.D.C. di Vimercati and P. Samarati, Authorization specification and enforcement in federated database systems, *Journal of Computer Security* 5(2) (1997), 155–188.
- [22] C. Ellison, SPKI/SDSI certificates, <http://world.std.com/~cme/html/spki.html>, Aug. 2001.
- [23] 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.
- [24] T. Finin, Y. Labrou and J. Mayfield, KQML as an agent communication language, in: *Software Agents*, J.M. Bradshaw, ed., MIT Press, Cambridge, 1997. <http://www.cs.umbc.edu/kqml/papers/>.
- [25] B. Gladman, C. Ellison and N. Bohm, Digital signatures, certificates and electronic commerce. <http://jya.com/bg/digsig.pdf>, April 1999.
- [26] D. Heimbigner and D. McLeod, A federated architecture for information management, *ACM Transactions on Office Information Systems* 3(3) (1985), 253–278.
- [27] R. Hull and G. Zhou, A framework for supporting data integration using the materialized and virtual approaches, in: *ACM SIGMOD '96*, ACM, Montreal, Canada, 1996, pp. 481–492.
- [28] IETF X.509 Working Group, Public-key infrastructure (X.509). <http://www.ietf.org/html.charters/pkix-charter.html>, 1998.
- [29] S. Jajodia, P. Samarati, V. Subrahmanian and E. Bertino, A unified framework for enforcing multiple access control policies, in: *SIGMOD '97*, Tucson, AZ, USA, 1997, pp. 474–485.
- [30] 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.
- [31] 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*, Rensselaerville, New York, 1995, pp. 43–60.
- [32] A.Y. Levy, A. Rajaraman and J.J. Ordille, Querying heterogeneous information sources using source descriptions, in: *Proceedings of 22nd International Conference on Very Large Data Bases VLDB '96*, Mumbai (Bombay), India, Morgan Kaufmann, 1996, pp. 251–262.
- [33] W. Litwin, L. Mark and N. Roussopoulos, Interoperability of multiple autonomous databases, *ACM Computing Surveys* 22(3) (1990), 267–293.
- [34] L. Liu and C. Pu, Distributed interoperable object model and its application to large-scale interoperable database systems, in: *Proceedings of ACM International Conference on Information and Knowledge Management (CIKM '95)*, 1995.
- [35] E. Mena, V. Kashyap, A. Sheth and A. Illarramendi, OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies, in: *First IFICIS International Conference on Cooperative Information Systems (CoopIS '96)*, Brussels, Belgium, IEEE Computer Society Press, 1996, pp. 14–25.
- [36] 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.
- [37] M. Olivier and S. von Solms, A taxonomy for secure object-oriented databases, *ACM Transactions on Database Systems* 19(1) (1994), 3–46.
- [38] R.L. Rivest, L. Adleman and M.L. Dertouzos, On data banks and privacy homomorphisms, in: *Foundations of Secure Computation*, R. DeMillo et al., eds, Academic Press, New York, 1978, pp. 169–177.
- [39] 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, 1997.
- [40] A.P. Sheth and J.A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, *ACM Computing Surveys* 22(3) (1990), 183–236.
- [41] V.S. Subrahmanian, S. Adali, A. Brink, R. Emery et al., HERMES: Heterogeneous reasoning and mediator system, <http://www.cs.umd.edu/projects/hermes/publications/publications.html>.

- [42] Z. Tari and G. Fernandez, Security enforcement in the DOK federated database system, in: *Database Security, X: Status and Prospects, Proceedings of the 10th IFIP WG 11.3 Working Conference on Database Security*, P. Samarati and R.S. Sandhu, eds, Como, Italy, Chapman & Hall, 1997, pp. 23–42.
- [43] A. Tomasic, L. Raschid and P. Valduriez, Scaling heterogeneous databases and the design of DISCO, in: *Proceedings of the International Conference on Distributed Computer Systems*, Hong Kong, 1995.
- [44] J.D. Ullman, Information integration using logical views, in: *Proceedings of the 6th International Conference on Database Theory, ICDT '97*, number 1186 in LNCS, Delphi, Greece, Springer-Verlag, Berlin, 1997, pp. 19–40.
- [45] N.R. Wagner, P.S. Putter and M.R. Cain, Encrypted database design: specialized approaches, in: *IEEE Symposium on Security and Privacy*, 1986, pp. 148–153.
- [46] G. Wiederhold, Mediators in the architecture of future information systems, *IEEE Computer* **25**(3) (1992), 38–49.
- [47] G. Wiederhold, M. Bilello and C. Donahue, Web implementation of a security mediator for medical databases, in: *Database Security, XI: Status and Prospects, Proceedings of the 11th Annual IFIP WG 11.3 Working Conference on Database Security*, T.Y. Lin and S. Qian, eds, Lake Tahoe, California, IFIP, Chapman & Hall, 1998, pp. 60–72.
- [48] G. Wiederhold and M. Genesereth, The conceptual basis for mediation, *IEEE Expert, Intelligent Systems and their Applications* **12**(5) (1997), 38–47.
- [49] L.L. Yan, T. Özsu and L. Liu, Accessing heterogeneous data through homogenization and integration mediators, in: *Second IFCS Conference on Cooperative Information Systems (CoopIS '97)*, Charleston, South Carolina, 1997, pp. 130–139.