A Hybrid PKI Model with an Application for Secure Mediation

Joachim Biskup, Yücel Karabulut
Fachbereich Informatik
Universität Dortmund
D-44221 Dortmund, Germany
{biskup,karabulu}@ls6.informatik.uni-dortmund.de

Abstract. For distributed computing systems, specification and enforcement of permissions can be based on a public key infrastructure which deals with public keys for asymmetric cryptography. We review previous approaches and classify them as based on trusted authorities with licencing and dealing with free properties (characterizing attributes including identities), e.g. X.509, or based on owners with delegation dealing with bound properties (including capabilities), e.g. SPKI/SDSI. These approaches are extended and integrated into a hybrid model which uses protocols to convert free properties into bound properties. Furthermore we unify licencing and delegation by introducing administrative properties. The hybrid model is suitable for a wide range of applications requiring security policies for confidentiality and integrity. In the latter case appropriate challenge-response protocols are needed. Secure mediation is taken as an example for such applications.

Keywords. distributed system; permission; public key infrastructure; trusted authority; licencing; free property; identity; owner; delegation; bound property; capability; administrative property; confidentiality; integrity; challenge-response; secure mediation.

1 Introduction

The proper administration of IT-systems requires to specify which clients are allowed to access which services, and to effectively and efficiently enforce such specifications. In a local system, a specification can be represented by traditional access rights granted to known identified individuals and thereby to the processes under their control, and enforcement is based on identification and authentification of requesting individuals over a trusted physical path and on keeping track of the processes they are controlling (see for instance access control in UNIX). In advanced *distributed systems* [22], however, some basic assumptions of this approach are not longer valid. In particular, a client may not be registered in advance by an identifying name at the site of a server, and thus he may be unknown to a server at the time of a request. And we cannot reasonably assume anything like a trusted physical path between remote agents. In order to overcome these and related difficulties a diversity of proposals has arised. While all proposals exploit cryptography, some of them use symmetric cryptographic mechanisms, like Kerberos [16], and others rely on asymmetric cryptography, like X.509 [13, 14] and SPKI/SDSI [7,12,5].

In this paper we deal with approaches to specify and to enforce *permissions* of clients on remote servers which are based on *asymmetric cryptography,* see for instance [6,4,3,19]. 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 attributes that are claimed to be true for a public key or the holder of the corresponding secret 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, like for example the European directive for digital signatures [8], over conceptual issues to final implementations. Here we concentrate on *conceptual issues*.

More specifically, we aim at designing a hybrid model for a PKI to be used for specifying and enforcing permissions in distributed computing systems. The hybrid model is intended to integrate and unify previous approaches. The design has been motivated by the requirements which we have seen for secure mediation, as reported in [1], but appears to be worthwhile for a broad spectrum of applications.

Roughly summarized, in a *mediated information system* [24,25], a client, looking for a specific piece of information, addresses a mediator asking for assistance in retrieving, homogenizing and assembling data from any sources the mediator may hold worthwhile to contact. In this scenario, clients and sources do not know each other; they even do not exchange messages directly. Nevertheless, the sources, as well as the mediators, 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. These requirements are met by our design of *secure mediation* [1], shortly outlined as follows.

A client proves his eligibility to see a piece of information by a collection of so-called personal authorization attributes. Independently of a specific request and a specific source, a personal authorization attribute has been assigned to the client by some trusted authority who certified such an assignment within a credential. Here a credential[1] is a digital document which, in the simplest form, contains a (representation of a) personal authorization attribute, a public key of the client to whom the attribute has been assigned, and the digital signature of the trusted authority for the document. A source always receives a mediated request to deliver some information together with a set of credentials stemming from the pertinent client. Then the source decides on the permission of the request by evaluating the credentials and the contained personal authorization attributes with respect to its *confidentiality* policy. In case of an allowance, the returned data is encrypted with the public keys found in the credentials on which the permission decision has been based. Thus the returned data can only be decrypted by that client who has proven his eligibility by showing an appropriate collection of personal authorization attributes.

We argue that similar scenarios can be found in distributed systems not only for mediation but for many other applications as well. The following points are crucial:

[1] Here we still use the term *credential* as in [1]. In the present paper, we would prefer to call the document a *certificate* in the sense of Section 2.4. However, the document is also used like a *credential* in the sense of Section 2.5. This dual use is further discussed in Section 5.

- A client is represented by (one of) his public key(s) and characterized by the assigned attributes.
- A trusted authority assigns attributes to public keys.
- A server follows a security policy that is expressed in terms of attributes.

Some of these applications may be concernend not only with confidentiality but also, or even exclusively, with *integrity*. In that cases, however, an additional problem has to be solved. At the time of a request, a client has to convince the server that he, the client, actually holds the secret key(s) corresponding to the submitted public key(s). Typically, the client has to correctly *respond* to a *challenge* generated by the server (where here the correctness criteria roughly says that the response results from a fresh usage of the secret key(s)).

The hybrid model for a PKI presented in this paper constitutes a far more detailed elaboration of the crucial points. The model is adaptable for security policies concerned with confidentiality and integrity if suitable challenge-response mechanisms are implemented. Figure 1.1 visualizes some of the details of the model to be explained in the following sections. The next Section 2 emphasizes the subtle distinctions between the usually hidden (real) world and its visible representation (Section 2.1), classifies properties of clients and servers (Section 2.2), discusses the role of administrative properties (Section 2.3), summarizes how trusted authorities deal with so-called free properties (which include personal authorization attributes) using licencing if needed (Section 2.4), and 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 (Section 2.5). Section 3 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 assembles the constituting parts into the full hybrid model. Section 5 reconsiders secure mediation. Finally, Section 6 reviews related work and presents conclusions.

2 Hidden (Real) World and Visible Virtual Views

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.

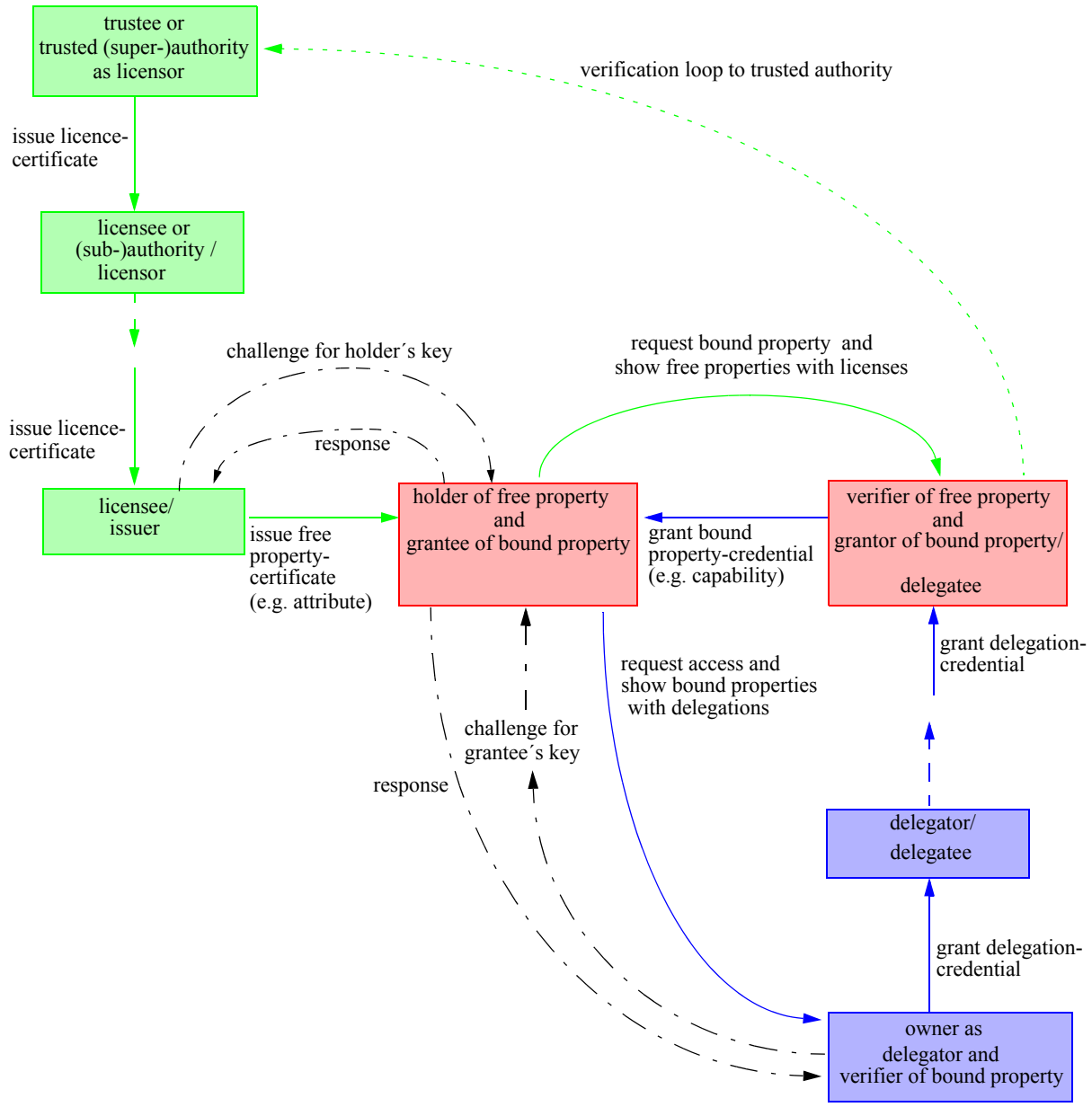

Figure 1.1 Outline of an instance of the hybrid model for a PKI

Surely, the entity can send and receive messages to and from this hidden (real) world, and based on these messages the entity can produce a virtual view which is actually visible to it. As a consequence, security policies and permission decisions are solely grounded on the locally

available visible view on the global (real) world. This sketched exposition is elaborated in the next subsections and visualized by Figure 2.1.

2.1 Property Assignment and Certificates/Credentials

An identifiable *entity* in the (real) world might be an *individual* or 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 *secret and public keys*: while the entity must keep the secret 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* 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;
- 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 secret key for signing.

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 how to process the document;
- a *validity* field which might indicate that the responsible entity has limited the property assignment to a certain time period or that the responsible entity has otherwise restricted the usability of the document.

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 has to 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 (private keys) appearing in the document representing the supposed entities?

The treatment of these issues is started in Section 2.3.

2.2 Properties

In the following we classify properties of entities (and thus contents of certificates and credentials) according to two aspects. The first aspect deals with properties which characterizes entities with security policies in mind. We distinguish two kinds of such *characterizing properties* (see Figure 2.2):

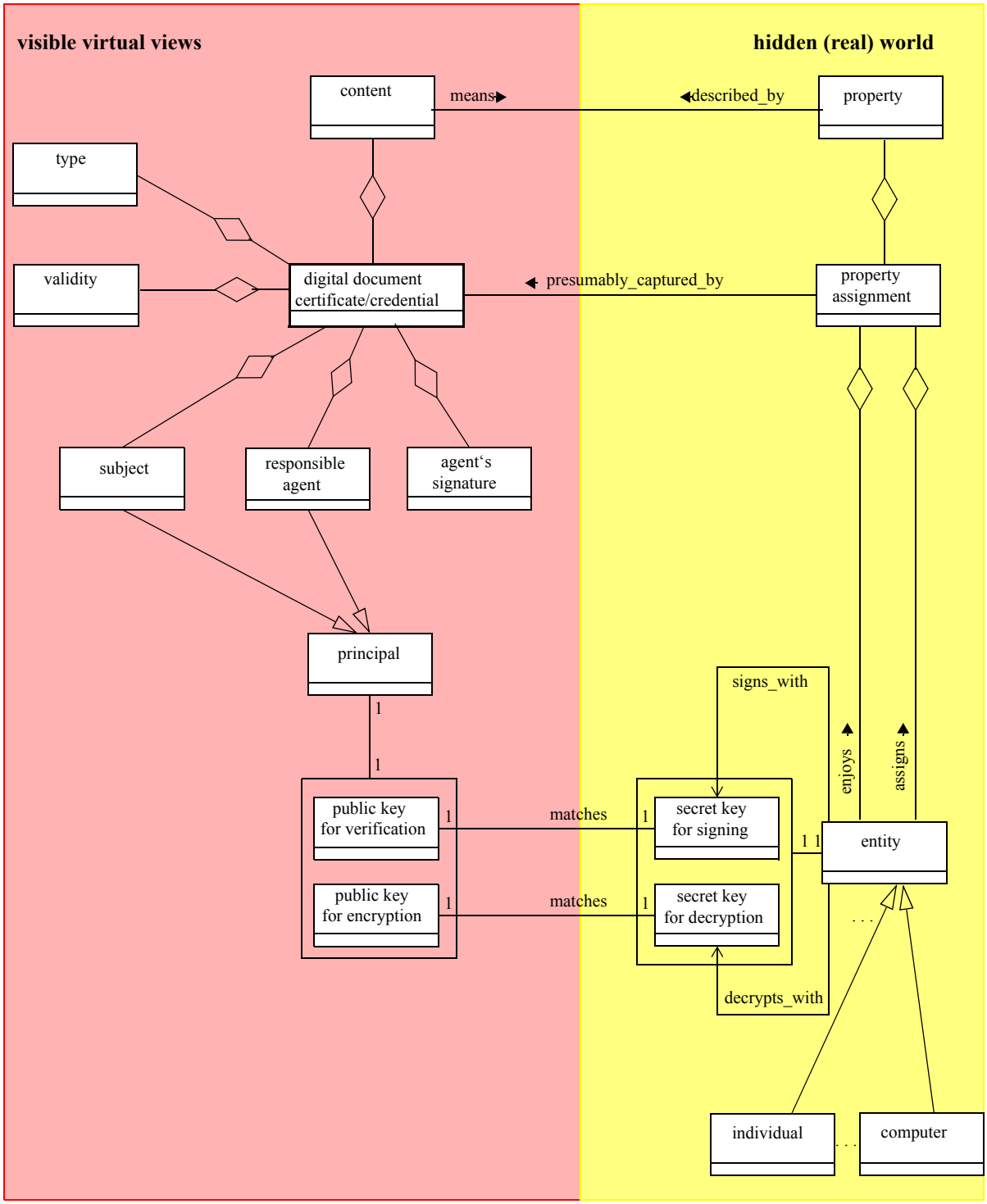

Figure 2.1 The hidden world of entities which assign and enjoy properties and its representation by documents (certificates and credentials)

- 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 base their security policies and permission decisions on shown free properties, but in general these entities 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, ...). 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 some kind of promise to get a specific service as expressed in the relationship.

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 how an entity can exploit its free properties in order to acquire bound properties, or speaking otherwise, how a client entity can convince a server entity to grant him the permission to use some services. This analysis is given in Section 3 and Section 4.

The second aspect deals with the administration of characterizing properties. More specifically, 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* (see Figure 2.3):

- 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 has to be suitably expressed, again by appropriate administrative properties.

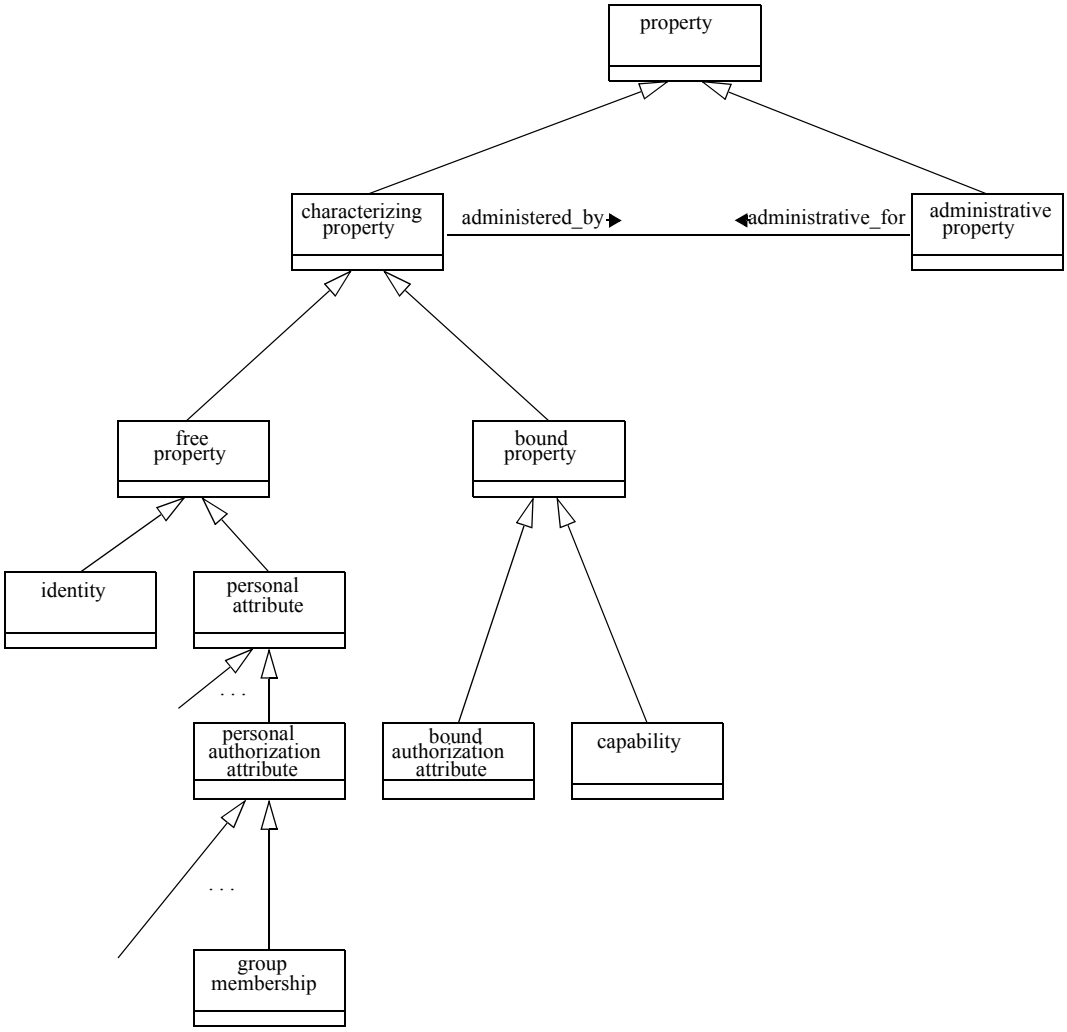

Figure 2.2 A coarse classification of characterizing properties

- The *administration 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 2.4 and Section 2.5 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

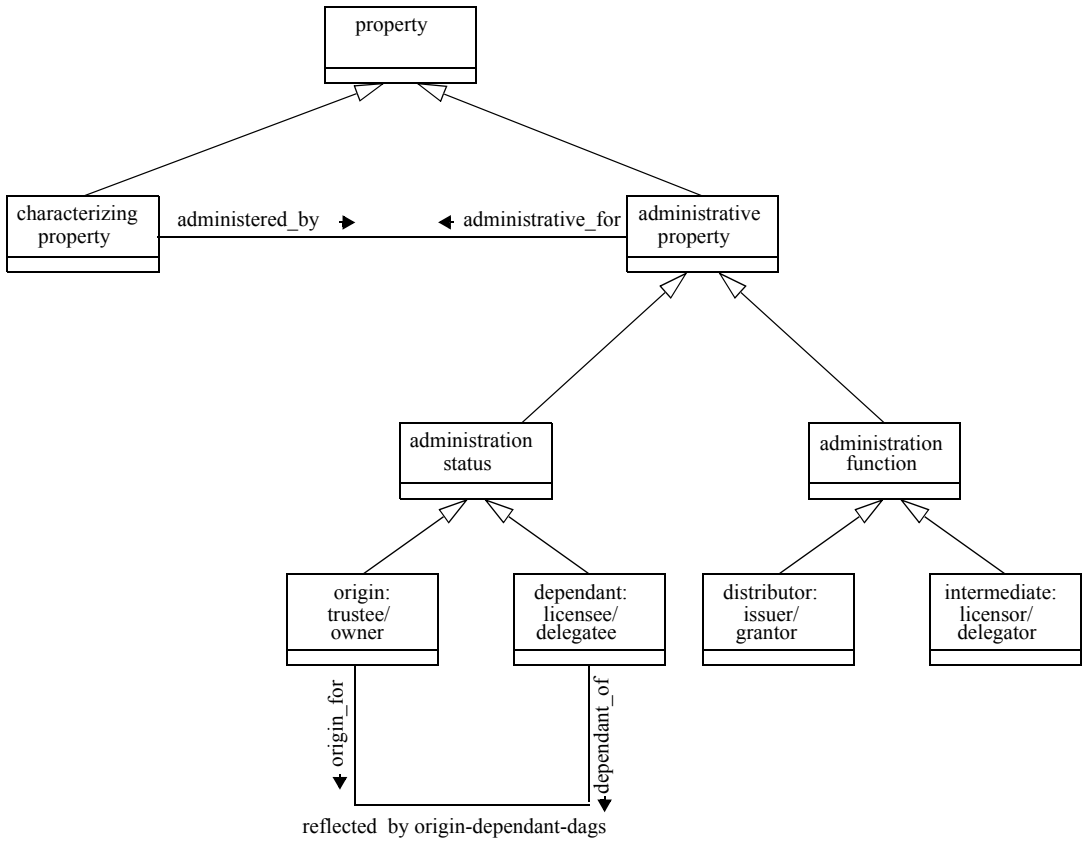

Figure 2.3 A classification of administrative properties

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) assignments of properties to entities, whether they refer to characterizing or administrative properties, have to be represented in the visible virtual world of certificates and credentials. Thus the specialization hierarchy for properties, as shown in Figure 2.2 and Figure 2.3, has to be suitably mirrored in a corresponding specialization hierarchy for certificates and credentials (which is not explicitly exhibited in this paper). 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.

2.3 Evaluating trust about ideal claims

At the end of Section 2.1 we noted that a specific entity has to evaluate his individual trust about the ideal claims concerning the relationships of *presumably_captured_by*. The very purpose of the *administrative properties*, 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 have to 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 valid indeed 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 meant by the content of the ‘‘main document’’.

In order to be helpful, the ‘‘main document’’ and its ‘‘supporting documents’’ should form a *directed acyclic graph* (*dag*) with respect to their relationsships concerning *support*. A corresponding class model is visualized by Figure 2.4 which can be understood as an extension of

the left part (about the visible virtual views) of Figure 2.1 devoted to formalize the borderline relationship *presumably_captured_by* strictly within the visible virtual views. As a special case, we just get a *chain*. For example, Figure 1.1 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 having 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 1.1 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.

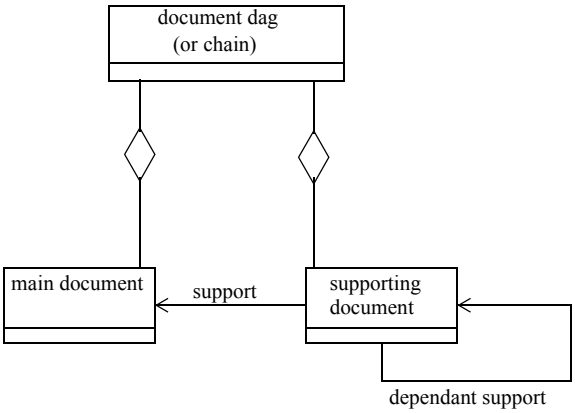

Figure 2.4 Structure of document dags (and chains) which formalize the borderline relationsships of *presumably_captured by*

2.4 Model of Trusted Authorities and Licencing

Following and extending the basic approach of X.509 [13,14], free properties and the corresponding certificates are handled by trusted authorities using licencing. 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 about this assignment. In the certificate, the content means 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 secret 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 certicate, 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 *licenso*r and trusting the *licencee*, 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 trustworthyness to assign the specific free property, the subject is the principal (public key(s)) representing the licensee, and the responsible agent is the principal (public key(s)) representing the licensor.

Additionally, licencing can be transitively organized: a licensor can express his trust into 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 has to be somehow 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.

In this model of trusted authorities and licencing, a special kind of certificates occur, namely the *identity-certificates* which deal with free properties that can be used to identify an entity.

Typically, such a free property is a name or some number uniquely used within a domain. Then in an identity-certificate, the content means the identified entity, the subject is the principal, i.e. the public key representing the identified entity, and the responsible agent is the principal representing the entity that certifies that the key occuring as subject belongs to the identified entity.

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 poperties. Again, also the direct issuer's identity might 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.

2.5 Model of Owners and Delegations

Following and extending the basic approach of SPKI/SDSI [7,12,5], bound properties and the corresponding credentials are handled by owners of services using delegation. 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.

In *acting*, such an entity assigns a bound property to another entity and, accordingly, *grants* a suitable 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 secret key for signing.

In *being addressed* as an owner, such an entity is afterwards contacted by some further entity that 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 meant 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 before 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. Then, basically, 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. In the delegation-credential, the content means trustworthyness 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 into 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.

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 of the services to which the bound property refers. Moreover, any dag of supporting delegation-credentials contains just one origin, namely the owner himself.

3 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 *security policy* that maps free properties on bound properties. More precisely, the policy specifies

which set of free properties an entity has to enjoy in order to get a bound property assigned.

Rephrased more technical in terms of the visible world of digital documents, this means the following: the security policy specifies

which free poperty-certificates as ‘‘main document’’ together with which ‘‘supporting licence-certificates‘‘ are accepted in order to get which bound property-credential granted.

The middle part of Figure 1.1 visualizes the situation. The entity on the right is the grantor following a security 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 security 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.

Traditional access rights management with capabilites or with access control lists can be seen as degenerated special cases which exploit identifying names as free property. In both versions an identifying name as 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. Also secure mediation, as sketched in the introduction, constitutes a special case where bound properties are not explicitly delivered.

As noted before, in general free properties need not to be identifying. Moreover, in distributed systems where entities want to cooperate without knowing each other in advance, identifying free 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 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 an 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 issuer or owner has to *challenge* the claiming entity to prove that he holds the matching secret key. Usually, the proof is accomplished by an appropriate *response* which is generated with the matching secret key. A more sophisticated version of authentification is achieved when the key holder performs a zero knowledge proof of knowledge (about the secret key) against the issuer or owner. Alternatively, if the issuer (but not the owner) is still thinking in terms of identities, he can also exploit an identification certificate with the considered public key as subject together with a supporting dag of licence-certificates.

It is important to note, that for the basic secure mediation as sketched in the introduction only authentification for issuing free properties is necessary. The reason is that here the properties are bound to a public key for *encryption*, and this key is used for encrypting the returned sub-answers. Accordingly, if free properties are certified for a properly authentificated key and confidentiality of the returned data is the only security concern, then there is no need to further authentification afterwards.

4 The Full Hybrid Model

4.1 Composing certificate processing, credential processing and their links

For the sake of succintness, we only summarize previous considerations without explaining additional details and special cases. An *instance* of the full hybrid PKI model consists of overlapping components of three kinds:

- trusted authorities (also called trustees) and licencees for and a holder of a free property (see Section 2.4),
- an owner and delegatees for and a grantee of a bound property (see Section 2.5), and
- a holder of free properties and a grantor of a bound property (see Section 3).

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 granteee of a bound properties *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 type. The link identifies the verifier of the former component with the grantor of latter component, and the holder of the first component with the granteee of the latter component. In some special cases a linked component may appear to be degenerated. This also allows to subsume traditional access right management under the hybrid model. The links are the important feature that makes the hybrid model highly flexible and adaptable for many applications. Figure 1.1 shows a simple example in which two loops for chains are linked.

4.2 An application

Though the two underlying models are quite different in their detailed requirements and their purpose for the hybrid model, there may arise situations which allow to use either of them, but notably with different semantics. As an example, consider the following situation. An entity tries to find out which other entities are willing to offer him a specific service and then requests it.

In a first version, the service seeking entity collects suitable free property-certificates and supporting licence-certificates and then sends a corresponding dag to some mediating entity together with the request for a bound property-credential for the wanted service. The mediating entity decides on the request according to his security 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 pro-

perty-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 the received dag.

In another version, the service seeking entity acts as an owner of his personal special mailbox for offered services. So in principle he can grant a bound property-credential as a capability that allows to insert an offer in his personal special mailbox. Rather than granting such a credential by himself (since he does not know to whom) 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. Using the corresponding chain the provider is allowed to insert his offer, and let us assume he does so. 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 police. In the positive case, he immediately executes the service seeked for.

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 Secure Mediation

We are using the hybrid PKI model to implement an extended version of our design of secure mediation [1]. The basic design, as sketched in the introduction, only exploited the PKI model of trusted authorities and licencing: We assumed that trusted authorities and their licencees for a special kind of free properties, called personal authorization attributes, were already in operation. And thus we had to enable clients, mediators and sources to act as verifiers of free properties, and mediators and sources had to define their confidentiality policies in term of personal authorization atttributes. So far, there were no explicit bound properties since mediators and sources executed permitted requests immediately.

The motivation to extend the basic design and to elaborate the hybrid PKI model presented in this paper has mainly originated from a conceptual challenge and a corresponding implementation task. Both the challenge and the task are related to our view that all entities are highly autonomously cooperating, and accordingly all entites are implemented as software agents which base their communications on KQML [9] and their data exchanges on CORBA [17,18].

The conceptual challenge concerns how remote and autonomous entities can agree on a common understanding of ‘‘personal authorization attributes’’. On the one side, trusted authorities assign personal authorization attributes as *free properties*, in principle without knowing their later usage. And on the other side, sources and mediators independently define their security policies in terms of personal authorization attributes, and thus implicitly treat them like *bound properties* (in a degenerate case of the model of owners without explicitly granting capabilites with the corresponding promise of a service). Surely, a client getting a free property assigned *would prefer* to look on the corresponding certificate as if it was a 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 owner of a source). Also clients wish to be assisted by a mediator to assemble appropriate properties within suitable digital documents in order to receive the information services they want. Thus mediators should participate in the common understanding of personal authorization attributes, too.

Inspecting the problem more thoroughly, however, we see that we can rarely postulate that autonomous entities in a worldwide information infrastructure can reach a common understanding implicitly. Rather it turns out that, in general, for any specific case we need an explicit *mediation process* that helps to interpret a free property as a bound property for some service and to arrange the corresponding conversion, as explained in Section 3. Then a mediating agent is acting both as verifier of free properties and as a delegatee and grantor of a bound property on behalf of a source which is the owner of an information service, as visualized by Figure 1.1. In the basic design of secure mediation, the sources themselves still took the burden of this task.

The implementation task concerns the software agents for secure mediation. For any singular mediation request, the specific entites directly involved perform a fixed role as client, mediator

or source, respectively, and those entities 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 our prototype implementation, 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.

Accordingly, we provided an agent PKI framework having the following main features:

- Generation of public/secret key pairs, keystore and self-signed X.509 certificates (we use the X.509 certificates created by Java *keytool* to get the public key stored in the digital certificate which is physically stored within the keystore).
- XML-encoding of free property-certificates and bound property-credentials.
- Issue of XML-encoded certificates and credentials.
- Verifying the signatures of 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).
- Validity checking for certificates and credentials.
- Content management operations for certificates and credentials including extracting the free properties and the bound properties.
- A GUI and command-line interface for administration of certificates and credentials.
- The PKI framework is written in Java and provides additionally an API for C++ applications and agents (e.g., our MultiMediaMediator agent).

Our implementation of the agent PKI framework is broad enough to allow to make experiments with different PKI approaches and a hybrid PKI model as described in this paper. We opted for an approach to certificates and credentials based on digitally signed XML documents in order to have freedom of experimenting and researching and not to be constrained by any of the existing certificate and credential formats. However, our approach is compatible with both

X.509 and SPKI/SDSI philosophies. We have used IBM's XML Security Suite for signing and verifying XML documents. The high-level API is designed to hide the complexity of the underlying security mechanisms but facilitate service requests through simple service calls.

The agent functional core includes functional modules to process and to evaluate the *content* of the KQML-performatives. What the content of an KQML-performative expresses depends on the specific agent in which the KQML-performative is implemented. For instance, the content for a MultiMediaMediator agent is an OQL query. The communication interface of the agent core includes now new KQML-performatives to handle the external actions and reactions among agents involved in the hybrid PKI model.

6 Related Work and Conclusions

Most of the works investigating the application of certificate/credential-based access control treat both PKI models discussed in Section 2.4 and 2.5 as competing approaches and base their work on a single PKI model. Even some of these works abstract from any particular PKI model.

In [20,26] the authors propose to use credentials in a Web-based client/server architecture. They present the tasks needed for credential management both on the client side and server side. They express access control rules in a logic language. Their work on credential management focuses on ‘‘credential acceptance policies’’. [27] investigates trust negotiation strategies and discusses credential disclosure policies. [21] presents the privacy vulnerabilities during trust negotiation. [2] introduces a credential-based formal framework and a model to regulate access and information release over the Internet. These works do not make any assumptions on the underlying PKI model.

[23] use X.509 based identity/attribute certificates and use-condition certificates (policy assertions) for access control. Use-condition certificates enable distributed stakeholders to share control over access to resources. In their architecture, at time of a resource access, the policy engine module gathers the certificates associated with a user to determine, if the use conditions are met.

[11] 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. This work focuses on X.509 certificates and concentrates on unifying different credential formats.

[15,10] presents a security architecture which has been implemented using SPKI-like authorization certificates. They focuse on authorization and delegation in distributed agent systems.

In [1] the authors discuss the security requirements for mediation and propose a general design of secure mediation. They also present a specific security architecture implementing the general design. The secure mediation approach has exploited the PKI model of trusted authorities and licencing.

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 this paper, 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 how trust evaluations are performed. We argue that many applications require to use and to link both kinds of PKI. Accordingly, we outlined a *hybrid PKI model* which unifies and extends the previous approaches. Thereby we introduced explicit *administrative properties* to be encoded in supporting documents, and we exhibited the *generic structure of links* between instances of the previous approaches. 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. The *core functionality* of the PKI hybrid model, as needed for an extended version of our approach to *secure mediation*, has been implemented using XML-encoded documents.

There are various topics for future research and development. First of all, several details of the the hybrid PKI model need further elaboration, including a formal specification and precise translations for embeddings of previous approaches. Based on these foundations, the prototyp 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.

7 References

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

[2] Bonatti, P., Samarati, P., Regulating service access and information release on the web, Proceedings of the 7th ACM Conference on Computer and Communication Security, Athens, Greece, Nov. 2000, pp.134-143.

[3] Brands, S.A., *Rethinking Public Key Infrastructures and Digital Certificates*, MIT Press, Cambridge-London, 2000.

[4] Chaum, D., Security without identification: transaction systems to make big brother obsolete, Communications of the ACM 28,10 (1985), pp. 1030-1044.

[5] Clark, D., Elien, J.-E., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L., Certificate chain discovery in SPKI/SDSI, Journal of Computer Security 9 (2001), pp. 285-322.

[6] Diffie, W., Hellman, M.E., New directions on cryptography, IEEE Transactions on Information Theory 22,6 (1976), pp. 644-654.

[7] Ellison, C.M., Frantz, B., Lampson, B., Rivest, R., Thomas, B.M., Ylonen, T., Simple public key certification, `http://www.ietf.org/ids.by.wg/spki.html`, 1999.

[8] European Parliament and Council, Directive 1999/93/EC on a Community framework for electronic signatures, 13 December 1999, Official Journal of the European Communities L13/12-20, `http://www.qlinks.net/comdocs/elsig/en.pdf`.

[9] Finin, T., Labrou, Y., Mayfield, J., KQML as an agent communication language, Software Agents (ed.: Bradshaw, J.M.), MIT Press, Cambridge, 1997, `http://www.cs.umbc.edu/kqml/papers/`.

[10] Helsinki University of Technology-Telecommunications software and multimedia laboratory, *Telecommunications software security architecture (TeSSA)*, `http://www.tml.hut.fi/Research/TeSSA/`, 1998-2001.

[11] Herzberg, A., Mass, Y, Relying party credentials framework, Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference (D. Naccache, editor), San Francisco, CA, 2001, Lecture Notes in Computer Science 2020, Springer, Berlin etc. , 2001, pp. 328-343.

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

[13] IETF X.509 Working Group, Public-key infrastructure (X.509), `http://www.ietf.org/html.charters/pkix-charter.html`, 1998.

[14] ITU-T Recommendations X.509: The Directory - Public Key and Atrtribute Certificate Frameworks, 2000.

[15] Lehti, I., Nikander, P., Certifying trust, 1st International Workshop on Practice and Theory in Public Key Cryptography, San Diego, CA, March 1998, IEEE, pp. 83-98.

[16] Miller, S.P., Neuman, B.C., Schiller, J.I., Saltzer, J.H., Section E.2.1: Kerberos authentification and authorization system, *M.I.T. Project Athena*, Technical Report, Cambridge, Massachusetts, 1987.

[17] OMG (Object Management Group), *Common Secure Interoperability*, OMG Document orbos/96-06-20, July 1996.

[18] OMG (Object Management Group), *CORBA Security*, OMG Document 96-08-03 through 96-08-06, July 1996.

[19] Samarati, P., de Capitani di Vimercati, S., Access control: policies, models, and mechanisms, FOSAD 2000 (Focardi, R., Gorrieri, R., eds), Lecture Notes in Computer Science 2171, Springer, Berlin etc., 2001, pp. 137-196.

[20] Seamons, K.E., Winsborough, W., Winslett, M., Internet credential acceptance policies, Proceedings of the Workshop on Logic Programming for Internet Applications, Leuven, Belgium, July 1997.

[21] Seamons, K., Winslett, M., Yu, T.,Smith, B., Child, E., Jacobsen, J., Protecting privacy during on-line trust negotiation, 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA, April E. 2002, to appear.

[22] Tanenbaum, A.S., van Steen, M., *Distributed Systems*, Prentice-Hall, Upper Saddle River, N.J., 2002.

[23] Thompson, W., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A., Certificate-based access control for widely distributed resources, Proceedings of the 8th USENIX Security Symposium, Washington D.C., Aug. 1999.

[24] Wiederhold, G., Mediators in the architecture of future information systems, IEEE Computer 25,3 (1992), pp. 38-49.

[25] Wiederhold, G., Genesereth, M., The conceptual basis for mediation, IEEE Expert, Intelligent Systems and their Applications 12,5 (1997), pp. 38-47.

[26] Winslett, M., Ching, N., Jones, V., Slepchin, I., Assuring security and privacy for digital library transactions on the web: Client and server security policies, Proceedings of Forum on Research and Technical Advances in Digital Libraries (ADL'97), Washington, DC, May 1997, pp. 140-152.

[27] Yu, T., Winslett, M., Seamons, K., Interoperable strategies in automated trust negotiation, Proceedings of 8th ACM Conference on Computer and Communication Security, Philadelphia, Pennsylvania, Nov. 2001, pp. 146-155.