An Access Control Mechanism for P2P Collaborations

Christoph Sturm, Klaus R. Dittrich, Patrick Ziegler
Department of Informatics
University of Zurich
{sturm,dittrich,pziegler}@ifi.uzh.ch

ABSTRACT

We introduce a fine grained access control mechanism for Peer-to-Peer collaborations. It is based on the local access control components of the participants. The peers export their access control policies in XACML. Two mechanisms are proposed to combine these policies. The first approach establishes mappings between the export policies. The second approach installs a distributed access control directory. While mappings are created between two peers, a directory contains all rights of all users of all peers. We compare these two approaches and discuss their pros and cons.

1. INTRODUCTION

A new interconnected and collaborative world is approaching. Many signs for this thesis are found in web 2.0 [15] labeled applications. Internet based, easy to use collaborative applications (flickr, wikis, youtube, to name a few) are getting more and more popular and show how powerful such applications can be. All these applications are centralized, but the real power of such applications would emerge if they were distributed in a Peer-to-Peer (P2P) manner. This means that every member stores his contributing content on his local computer and links it to the data of the other participants. In this way, we get a real collaborating network where not only the content but also the service delivery is collaborative. In addition, the idea of dataspaces [9] comes into play. Loosely connected databases can decide themselves how tight the connection to the other data sources should be. Furthermore, the connections can be established incrementally in a "pay as you go" fashion. In such environments a decentralized access control system can be of great value. Although the decentralized approach is not mandatory, it is the most flexible way to connect several data sources with each other. Examples for dataspace scenarios are personal information management, scientific data management and structured queries and content on the WWW. However, to bring P2P based collaborations into life, a security and access control framework is indispensable. P2P networks change the requirements for access control. The lack of a central authority is one of the basic features of P2P networks and this is the reason why traditional well known access control approaches cannot be applied. Nevertheless, there are many applications based on P2P technology that can benefit from an access control system, such as Peer Data Management Systems (PDMS) and other data exchange applications that are solely restricted to the processing of uncritical impersonal data like music and video.

To point out the need for such an access control framework, consider the following scenario. Assume that several research groups want to exchange parts of their scientific data. To do this they establish a P2P network. Because not every member of the collaborating research groups is allowed to access all data, they need fine grained access control restrictions on the shared data.

In this scenario, the research groups can alternatively set up a shared central server and negotiate the responsibilities and general usage conditions. Compared to the P2P approach this means more organizational effort, as the responsibilities and duties of the participants must be negotiated and established. Besides, it remains unclear if everybody can agree that his research data is stored on a remote server. Another crucial point is who owns the collaborative generated data. Is it owned by all participants or by the creator of the individual part of the content? In the P2P case this question can be easily answered, because the data, if we abstract from the use of replication for performance and reliability reasons, is stored locally so that the creator remains the owner.

Our contributions are as follows. We introduce a fine grained access control mechanism which enables peers of a P2P network to establish a global, network wide access control component. This is done in a collaborative manner without the need of a central authority. Moreover, the autonomy of the participants is not affected. They can either maintain the full control over their data or have the opportunity to delegate clearly defined parts of their control to other contributing peers and users. We propose two solutions to establish such a global access control mechanism. The first approach is based on mappings. The second approach establishes a distributed access control directory. We compare the two approaches and show their pros and cons.

The paper is organized as follows. In Sect. 2 existing work in our field is presented. In the next two sections we introduce our approach and describe the use of XACML policies. In Sect. 5 we present two approaches to connect the local access control components. After the approach of the dis-

DAMAP'08, March 25th, 2008, Nantes, France.

tributed access control directory is described in detail in Sect. 6, we present our access control data model in Sect. 7. Next pros and cons of the two approaches are discussed and finally the conclusion is presented in Sect. 9.

2. RELATED WORK

General security problems in P2P systems are discussed by several authors (e.g. [1, 8]). Their work lays the foundation to build access control mechanisms. One can consider the research on access control in federated databases [12, 7] as the basis for our work. Especially the work of Heimbigner and McLeod on loosely coupled federated databases [11] is of relevance, but the proposed mechanism is insufficient for our requirements. First, it is coarse grained because it is based on peers and not on users. Second, it depends on poorly scaling access lists attached to each object.

Recent papers consider access control in P2P networks. Miklau and Suciu [13] present an access control mechanism based on encryption and key distribution for simple P2P data and/or file sharing applications. The approach from Sandhu et al. [18] goes one step further. They use trusted computing technology to enforce the privileges on the client. In that way, not only the access to the data is protected but also the use of the data can be restricted. Berket et al. [1] use the Akenti [21] mechanism to realize the fine grained access control which relies on some centralized knowledge. In contrast, Crispo et al. [5] use eXtensible Access Control Markup Language (XACML) policies [3] to specify the access restrictions on the data offered by the peers. These policies are evaluated and stored at supernodes. This restricts the use of the approach to hierarchical unstructured P2P networks. Also the related approach from da Silva et al. [6] uses XACML policies. In case of a relatively static organization of the P2P network, there is no need for a central policy catalog. Tran et al. [22] relate the privileges of the participants to their individual trust values.

However, none of these approaches considers existing access control components and information residing on the peers. Moreover, administration distribution is not included in these approaches. We consider this as the key feature for the scalability of access control administration. As the number of users grows it must be possible to delegate and distribute the burden of assigning privileges to them.

3. THE ACCESS CONTROL MECHANISM

The goal of our work is to establish a global fine grained access control mechanism similar to those already available in relational database management systems. In particular, this means that users can grant each other object privileges and administrative privileges. These administrative privileges make it possible to authorize remote users to grant privileges to other users.

To achieve this in a flexible P2P manner, we establish mappings between access control components of the local data sources. This is done in such a way that a virtual global access control component results. Ideally, the distributed nature of the access control mechanism should be transparent for the end user. For this purpose, the contributing peers must fulfill several preconditions that are explained in the following section.

3.1 Assumptions

Our approach intends to connect servers with dozens of users to a P2P network as it is the case in a PDMS or in our scenario from the introduction. We do not connect thousands of home office PCs with limited bandwidth and unreliable connection times. First, the additional costs to establish an access control mechanisms are only justifiable if the contributing peers plan to stay in the network for a longer time. Second, such networks are normally highly dynamic. To cope with the dynamics, these systems make use of massive data replication. Replication means that a peer copies its data to another peer and gives it the full authority over the replicated data, which includes the enforcement of the privileges concerning this data. For information requiring access control mechanisms to protect it, this necessitates a lot of trust between the peers, which is again very unlikely to appear in such dynamic environments. For this reason, our approach does not consider replication at all. From an access control perspective replicates can be considered as separate data items which need no special treatment.

On this foundation, we can expect that a participating peer has already a local access control component to control access to its content. In addition we assume that participants offer their data in XML because linking several data sources works best with data that is at least semi-structured.

Another important point is the security infrastructure our access control mechanism is based on. The basis for access control is secure authentication of users and peers. Therefore, we establish a public key infrastructure (PKI) with certificates. Every user and every peer that wants to participate in the network needs a certificate signed by a central certification authority. This guarantees that the peer and user IDs are unique and that every peer can only have one identity. This enables us to protect the P2P network against "Sybil" [8] attacks. Beside secure authentication, the PKI infrastructure can be used for secure communication between both peers and users.

3.2 Local Access Control Components

We adapt the method known from PDMSs of establishing mappings between data sources. Unlike data mappings, our mappings do not translate between different data schemata but convert access control information, residing in the local access control components of the participating peers. The idea is now that every peer exports parts of its access control information in an agreed format. Afterwards, the participating peers can establish mappings between their access control data. This situation is shown in Fig. 1. Such a

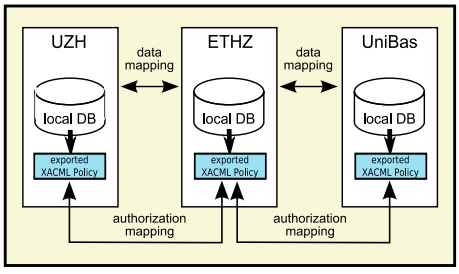

Figure 1: Collaborating peers with authorization mappings

relationship between local access control data results in a flexible connection of the access control components among the peers. It is up to the individual peers to whom they want to establish links and how tight the connection between their access control components is (for further details see [20]).

There are two levels of access control administration in our approach. The first level is the *peer level* which is managed by the administrator of a peer. In a nutshell, this level consists of the export policy of each peer. The export policy states which local users are allowed to participate in the P2P network, and what rights these users have over the local objects. The second level is *user based*. According to the rights of the individual P2P network users, privileges can be granted further to remote users. Provided that a user from ETH Zurich owns the required rights, she can, for example, grant the right to access an object of ETH Zurich to a user from the University of Basel.

If indirect mappings between the peers are allowed, which depends on how the data processing is done in the network, one can also grant indirect access privileges. In our example, the user at ETH Zurich can then grant access rights to a user at the University of Basel for an object at the University of Zurich. Despite the use of indirect mappings, each peer in our approach still has complete authority over its data. It can revoke or alter the granted privileges at any time.

To support data sources with various access control models, we need an independent, open and flexible export schema for the access control information, such as the XACML [3]. XACML is described in the next section.

4. XACML ACCESS CONTROL POLICIES

XACML is an OASIS standard for access control policies. A policy consists of PolicySets which aggregate other PolicySets or Policy elements. The basic components of a Policy element are Target and Rules. The Target specifies the application range of the policy. A Rule element determines the conditions to be checked. Each Rule can contain several Subject, Resource and Action elements. A Subject element specifies the attributes concerning the subject. The Resource element describes the conditions for the resources, while the Action element determines the actions the subjects are allowed to take on the resources.

If a user wants to access some data, policies concerning this request are evaluated and an access decision is made. Such an approach is very powerful, because different policies can be combined and various predicates can be analyzed during the evaluation. This expressive power and the open structure of XACML make it the preferred choice for the export format of access control information.

4.1 XACML as a Mediated Schema

Using XACML as an export schema means that every peer must express its access control policies of the shared data in XACML. Ideally, the participants have already an internal access control component based on XACML. In this case, the XACML export policy can be derived from the existing internal ones. Note that we need an export policy because there are additional restrictions for people accessing the data from outside, and not every local user is allowed to participate in the P2P network. Besides, it is a security threat to make access control information public. Thus, it is good practice to export as little information as possible. In fact, there is, if at all, only the need to show the policy to users who have access to objects residing on that peer. Additionally, converters for the most popular data sources and their access control mechanisms can be provided to ease the creation of export policies. The main problem for such converters is not to export information in XACML format and select items that should be exported, but to keep the content consistent. In particular, changes to local access rights of a data source must trigger an update on the exported XACML policy. In addition to serving as a mediator, the export policy has an important second role: it defines the P2P users of the individual peer.

Translating the access control information into XACML policies is straightforward. To support role based access control (RBAC) [14], we use the RBAC specification of XACML [2]. Users and roles are linked to XACML subjects, objects to XACML resources, and actions to XACML actions. The assignment of roles to users is done via an additional policy. A simplified example for user *hans@ethz* at *ETHZ* having read access to the object *ethz.object1* at *ETHZ* is shown in Fig. 2.

```
<Rule RuleId="export:ethz" Effect="Permit">
<!--
   <Grantor>
      <Subject>
         <SubjectMatch MatchId="&function;rfc822Name-equal">
            <AttributeValue
               DataType="&datatype;rfc822Name">admin@ethz
            </AttributeValue>
            <SubjectAttributeDesignator
               AttributeId="&subject;subject-id"
               DataType="&datatype;rfc822Name"/>
         </SubjectMatch>
      </Subject>
   </Grantor>
   <RuleGrantOption DataType="&xml;string">yes</RuleGrantOption>
   <Timestamp></Timestamp>
-->
   <Subject>
      <SubjectMatch MatchId="&function;rfc822Name-equal">
         <AttributeValue
            DataType="&datatype;rfc822Name">hans@ethz
         </AttributeValue>
         <SubjectAttributeDesignator
            AttributeId="&subject;subject-id"
            DataType="&datatype;rfc822Name"/>
      </SubjectMatch>
   </Subject>
   <Resource>
      <ResourceMatch MatchId="&function;anyURI-equal">
         <AttributeValue
            DataType="&xml;anyURI">ethz.object1</AttributeValue>
         <ResourceAttributeDesignator
            AttributeId="&resource;resource-id"
            DataType="&xml;anyURI"/>
      </ResourceMatch>
   </Resource>
   <Action>
      <ActionMatch MatchId="&function;string-equal">
         <AttributeValue
            DataType="&xml;string">read</AttributeValue>
         <ActionAttributeDesignator
            AttributeId="&action;action-id"
            DataType="&xml;string"/>
      </ActionMatch>
   </Action>
</Rule>
```

Figure 2: Excerpt of an export policy with added administrative instructions

4.2 Administrative Distribution

Privilege granting is not specified in the XACML frame-

work. In particular, it is not stated who is allowed to alter, delete or create a policy. The answer to this question is outside the scope of the XACML specification. Rissanen and Firozabadi [17] made suggestions how to include the administration of a policy into the policy itself. But only recently a working draft of XACML v3.0 [16] including some of their ideas was published. In this working draft just the creation of policies is controlled. Partial policy modifications are not supported. Thus the translation of the administrative information into XACML policies is not apparent. As we will see later, this information is vital in our collaborative approach, and we develop two proposals to address this issue.

The easiest solution to handle administrative distribution is to partition the policies into privileges. Every privilege corresponds to one policy. Then we can utilize the expressive power of XACML by using the XACML administration policies for administration distribution. The drawback of this is that the number of policies explodes fast, as we already have special policies for user role assignment. Consequently, we would need to split these policies and create for every piece an administration policy of its own. As this makes the creation and administration of export policies very cumbersome, we discard this possibility.

An alterative approach of granting privileges is to transfer the mechanisms valid for policy administration to the rule level. In this case, however, there are several open questions concerning the composition of the administrative policies. Most notably, there must be rules that reference the concrete rules in the policies and enforce conditions how a new rule can be created. This is too complicated for our approach, as it would result in an extention of the XACML policy schema and a change in policy evaluation. Thus we reverse the process. Instead of specifying who is allowed to create a new policy with what content, we just specify who is allowed to copy a specific rule and replace the subject or add a new subject (user/role) to a rule. On the one hand, this reduces the flexibility of the XACML administration policies. On the other hand, we avoid having to gather all the policies, evaluate restrictions and analyze conditions. We just have to check the authorization and evaluate the local policy with a new subject. Especially in our setting where we want to support collaboration between users, this can be regarded as a good compromise between flexibility and convenience.

To realize this solution we need three additional attributes at the rule level of the XACML attributes: the grant option <RuleGrantOption>, the grantor <Grantor> containing an XACML subject, and the timestamp <Timestamp>. To add these attributes at the rule level is a violation of the XACML specification. Admittedly, this information can be represented as an annotation to the XACML policy which is not relevant for the XACML policy evaluation. The task of the annotated attributes is to document and manage the authorization graph which will be checked by the P2P client on the peer. We therefore include these annotation attributes into an XML comment that is not considered by the XACML evaluation. For each peer, an export policy is generated by a single user with the corresponding rights for the whole database (e.g., the database administrator). Note that this administration information inside the export policy is only relevant to the outside world. That is, the export policy is always an extract of the local access control information, and local administration decisions are made on the local access control information. If changes affect privileges in the export policy, they are propagated to the export policy.

5. XACML POLICY INTEGRATION

After having established an agreed access control exchange format based on XACML, we need to exchange this information to create global distributed access control. The interconnections between exported XACML policies are defined on object level permissions. For instance, subject *hans@ethz* from peer *ETHZ* is allowed to access object *uzh.object4* at peer *UZH*.

Individual users can give other participants the right to access individual objects. This is the same as in a local situation. The interconnections are always between a subject of the one peer and the object of the other. There are two ways how to establish these interconnections, namely by the creation of mapping policies or the introduction of a distributed directory. Both approaches are presented below and will be further discussed in the following sections.

5.1 Mapping Policy

The first solution to establish connections between subjects and objects at different peers, called mapping policy (MP), is based on mappings. The participants create contracts between the diverse XACML export policies to get access to each others' data. Such a mapping is again an XACML policy which contains all the access rights regarding the contractual partner. The contract (i.e., the XACML policy) is stored at both contractual peers. They are responsible to protect this policy against unauthorized changes. In contrast to export policies this is crucial here, because the mappings are established in a collaborative way and can be altered by users of the contractual partners. An example of such a network is shown in Fig. 3. Every peer has an export policy and mappings exist between ETHZ and UZH, as well as between UZH and UniBas. In this situation the request is processed as follows. First the requester *hans@ethz* sends a request for object *unibas.object7* to the owner of the object *UniBas* (1). The peer *UniBas* then checks the authorization by asking all linking peers for the relevant mapping policies (2). In this case *UZH* is asked for the mapping policy for *ETHZ*. The mapping policy is returned to *UniBas* (3). *UniBas* evaluates the policies and informs the requester about the decision (4).

5.2 A Distributed Directory

Besides the mapping policy approach, the second solution to the problem of establishing connections between subjects and objects on different peers is to establish a distributed access control directory (DACD) with all the roles, users, objects and rights. If a privilege is granted to a remote user, it is stored in the distributed access control directory. In addition, these privileges are stored in the local repository of the grantor as a backup. Beside performance benefits, such a directory reduces the number of mappings needed and makes the integration process more transparent. Compared to the mapping policy approach, this results in a faster and more comfortable solution.

Fig. 4 shows the same scenario as in the mapping policy case in Fig. 3, with the access control processing in Fig. 4 using a distributed directory. Here, user *Hans* first sends an access request to the owner of the object *unibas.object7* at *UniBas* (1). The owner checks the authorization graph by consulting the distributed access control directory (2).

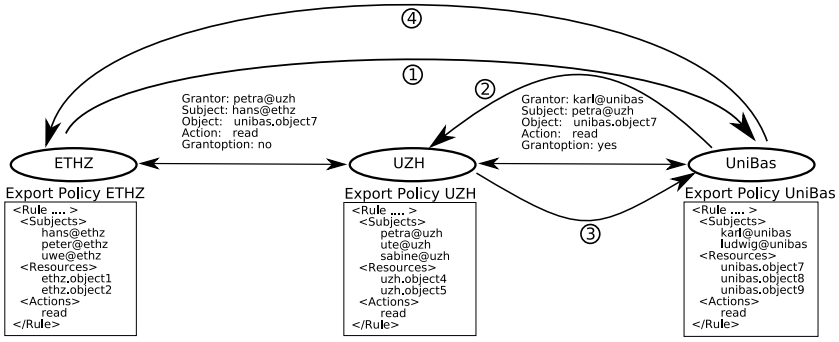

Figure 3: Mapping policy example hans@ethz requesting object unibas.object7

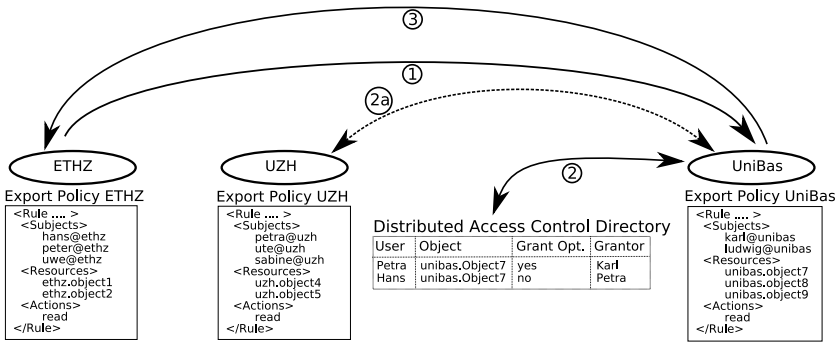

Figure 4: Distributed access control directory example for Hans requesting object unibas.object7

Peer *UniBas* can optionally inquire the correctness of the directory entries, labeled as (2a). Afterwards, the decision is returned to the requester (3).

In the following, we analyze the distributed access control directory in more detail, before we discuss the pros and cons of the two different approaches (distributed directory vs. mapping policy approach) regarding our requirements.

6. A DISTRIBUTED ACCESS CONTROL DIRECTORY

A distributed access control directory is functionally equivalent to an access control directory in a centralized architecture. Thus, it is unnecessary to create and maintain a separate infrastructure to express the relationships between export policies. Indeed, such a directory can be used to store the whole export policy content of every peer, which makes the export policies redundant. Every user and role is stored there with the corresponding privileges. But such a system-wide directory has important drawbacks. Many entries in the directory will never be used by most peers and hence only inflate the directory. Even worse, while we can restrict access to the export policy, entries in the directory can be seen by all participants of the network. In our approach, we only insert privileges into the directory which are granted to remote P2P users and retain the distinction between a mapping directory and an export policy. Just the mapping information that enables the combination of the local access control components is added to the directory. Unfortunately, the two levels (export policy and DACD) make it more difficult to decide if access should be granted. Beside the information in the directory, the decision maker needs the export policy to make a final decision.

Before we go into details how entries of this directory look like, we present the technology to store such a directory.

6.1 Storing an Access Control Directory

Storing the access control directory on a central server, as usually done in centralized systems, is not possible in the P2P case. P2P distributed directories offer the same basic services as a central directory. It is possible to store and find an object according to a key. P2P directories are scalable, fault tolerant, distributed over the peers and based on a distributed hash table (DHT). Beside the many advantages of this technology, in our application it has one drawback: an entry may only be correct for a specific point in time. Because a DHT guarantees no consistent state of the directory, we need to perform a consistency check with the result obtained from the directory.

Finally, a procedure must be defined that manages the actions to be taken after an entry disappears from the distributed directory. Connected entries can be removed or altered according to new situations. In the following, two possible DHT based directories are introduced.

6.1.1 PIER

In the PIER approach [10] a DHT overlay network is used to find and distribute content. The hashtable exists only in volatile storage, but its table portions are stored on non volatile storage at the participating nodes. The main emphasis of the approach is on distributed query processing.

But for the purpose of querying access control information we need no highly specialized query processing. To get the concerned data and scan it locally is thus sufficient for our approach.

6.1.2 Chord

Chord [19] stores each data item together with a key, which can be used to find the peer that stores the particular data item. Due to its simplicity, Chord can be used more flexibly and does not imply much overhead for the maintenance of the directory. For these reasons we opt for Chord as an underlying infrastructure for our access control directory.

6.2 Content Protection

Concerning the protection of the directory content, there are two objectives. First, all entries need to be protected against unauthorized changes. This can be achieved by signing every entry by its grantor. Entries with invalid signature can be removed from the directory. Second, security issues arise if all users can see the content of the whole directory in clear text. One can use encryption and private key distribution to solve this problem. Every object has a public key, and every entry in the directory for a certain object is encrypted with that public key. The corresponding private key is only distributed to authorized subjects that can therefore see the content of the entry. As long as the content of the directory changes infrequently, this is a practicable solution. Yet, in case of a revocation, all entries concerning the object must be reencrypted with a new key and the key must be redistributed. To avoid such a distribution of keys, entries can be encrypted using the public keys of the grantor and the grantee. In this case users can only see encrypted content which makes it harder to check the authorization state of a request. If indirect authorization mappings are possible, the intermediary peers must be consulted to decrypt and check the corresponding entries.

6.3 Directory Protection

In a central access control directory there are strict rules on who is allowed to access which entries in the directory. The same rules should apply in a decentralized case. By contrast, if the directory is distributed and a central authority is missing, the rules cannot be enforced. Although all participants have software managing the directory, it remains in the responsibility of each peer to test the correctness of the directory entries. This means that the correctness of the entries in the authorization chain have to be tested by every peer for each request.

DHT in general and Chord in particular have some security problems [23], especially with regard to security relevant data. How can we make the DHT secure in a way that no malicious peer can add, change or delete entries? At first, every user who wants to add, delete or update an entry has to authenticate herself by a challenge response authentication. Secondly, only the grantor can update or delete his own entries. However, malicious peers responsible for some part of the DHT can still conceal correct entries or resist to change, add or delete entries. Digging deeper into the DHT technology, the malicious peer can misroute messages and jam the whole directory. These problems of the DHT can be solved by secure routing primitives, as presented by Wallach [23]. In addition, we can detect malicious peers due to the fact that the directory at the originating peer, which consists of all entries it has granted to remote peers, differs form the DHT entries of the malicious peer. If we exclude the detected malicious peers, it is possible to achieve the required minimum ratio of 70% benevolent peers needed to guarantee secure routing according to Wallach [23]. Because the authorization information inside the entries is highly security relevant, it might be too dangerous to rely on the directory entries alone. Instead of trusting the directory, the owner can contact every grantor of the entries relevant for a specific request and check the validity of the entries.

7. THE GLOBAL ACCESS CONTROL DATA MODEL

After having presented where and how the connections between subjects and objects on different peers can be stored (mapping policies or distributed access control directory), this section focuses on the access control information itself. We discuss which information we need to store and how this information is represented in the directory and the mapping approach.

The authorization information for our global access control model is represented according to the conceptual schema shown in Fig. 5. This conceptual schema can be transformed into relations for the directory approach or into specific attributes in an XACML file.

We build on the common RBAC model, hence subjects can be a role or a user. Administrative delegation is done via standard grant and revoke mechanisms with cascading revocation. That is, if an object privilege is revoked from a user, all privileges for that object this user has granted further are also revoked. A role can own privileges even with grant option. Role membership can be granted through users who have the corresponding right or who created the role. To record the authorization graph we must store a timestamp for each privilege. We replace the timestamp with two local counters to avoid clock synchronization problems. Every user has a local grant counter that is incremented with every grant he issues. By storing the grant counter of the grantor and the grantee at the time of the grant we can construct the authorization graph. As a result, cascading revoke is supported. A cheating grantee can be immediately identified by the grantor through a lookup of counter values. The grantor is allowed to change the whole entry in any case, thus there is no security problem.

7.1 Directory Approach

In the directory approach the access control data is stored in relations. Therefore, we derive from our conceptual schema shown in Fig. 5 the two relations *Privilege* and *MemberOf* for our physical design. To distinguish between users and roles, we add a flag isRole to the Privilege relation. In Tables 1 and 2 we show the relations and corresponding sample data. “Gr” stands for grantor, “Ge” for grantee, and “P” for peer.

The distribution of privilege administration can generate a chain of privileges. To make the distribution more robust, one can automatically resolve the chains to a direct relationship between the owner of the object and the user who owns a privilege for this object.

7.2 Mapping Approach

In the mapping approach, the information of the global access control model (see Fig. 5) can be easily represented

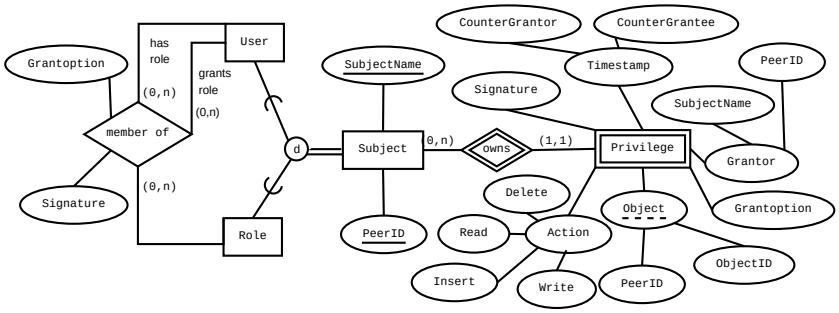

Figure 5: An EER diagram of the global access control data

MemberSubj.Name	MemberPID	RoleSubj.Name	RolePID	GrSubj.Name	GrPID	Grantoption	Sig.
uwe	ethz	readall	unibas	ludwig	unibas	no	sig

Table 1: The MemberOf table

in the export policy schema explained in Sect. 4. As stated in Sect. 5.1, the content of the mapping policies must be protected against unauthorized modification. Because the mapping policies are stored at the peers they belong to, the P2P client software is responsible for their protection. It should only be possible to modify the policy according to the authorization rules specified in the policies. To enable all P2P users to verify the correctness of the policy entries, we expect every rule to be signed by its grantor. In addition, the whole policy is signed by the last user who modified it. We use enveloping XML signatures to sign the entries, as specified in [4].

Each policy is controlled by the peer who granted the rights. Therefore it is in the peer's own interest to guarantee that the policy is correct. Note that mapping policies always establish direct authorizations between users of different peers. An example of excerpts from a rule of a mapping policy is shown in Fig. 6. It states that *uwe@uzh* at *UZH* is allowed to read object *unibas.object8* at *UniBas*. The right was granted by *ludwig@unibas* at *UniBas* without grant option.

```
<Rule RuleId="mapping:unibas:uwe:uzh:object8" Effect="Permit">
<!--
   <Grantor>
      <Subject>
         <AttributeValue
            DataType="&datatype;rfc822Name">ludwig@unibas
         </AttributeValue>
      </Subject>
   </Grantor>
   <RuleGrantOption DataType="&xml;string">no</RuleGrantOption>
   <Timestamp>
      <CounterGrantor DataType="&xml;integer">1</CounterGrantor>
      <CounterGrantee DataType="&xml;integer">0</CounterGrantee>
   </Timestamp>
 -->
   <Subject>
      <AttributeValue
         DataType="&datatype;rfc822Name">uwe@uzh
      </AttributeValue>
   </Subject>
   <Resource>
      <AttributeValue
         DataType="&xml;anyURI">unibas.object8</AttributeValue>
   </Resource>
   <Action>
      <AttributeValue
         DataType="&xml;string">read</AttributeValue>
   </Action>
</Rule>
```

Figure 6: A rule from the mapping policy between UZH and UniBas

8. DISCUSSION

Because we opt for simple rule policy administration, the directory and the mapping approach differ with respect to the management and recording of the administration privileges of remote users. The *directory approach (DA)* makes only sense if there is an open directory without encrypted entries. In this way everybody can see the rights of the users and roles. Besides the security threat that such an approach implies, the open directory assists other users in participating in the network. For example, assume that a role *assistant* is already established in the network. Before this role membership can be granted to a new assistant, the grantor and the grantee must know that such a role is already in existence. Since no network wide policy directory exists in the *mapping approach (MA)*, it is more difficult to find out which peer can grant certain privileges.

The *MA* tends to create many small mapping policies that form long authorization chains. With increasing authorization chain length an authorization becomes more volatile, since it can be destroyed by vanished chaining peers. Furthermore, the creation of these small mapping policies is very cumbersome. In the *DA* directory, entries remain valid even if a peer is disconnected as long as content and signature of the entry are not out of date.

For privilege enforcement the data provider has to check the authorization path which is provided by the requester. In the *MA*, all peers of the authorization path must provide their corresponding mapping policies to be able to come to a decision. On the contrary, the authorization path can be checked in the *DA* by retrieving the concerned entries from the distributed directory. Therefore, the communication overhead depends on the DHT and the underlying overlay network and not on the authorization path length.

The *MA* has a security benefit as access to the policies is protected by the peers that store them. It is in their

Subj.-Name	PID	is-Role	R	D	I	W	Obj.-PID	Obj.-ID	Grant-opt.	GrPID	Gr-Subj.	Gr#	Ge#	Sig.
uwe	uzh	no	yes	no	no	no	unibas	obj.8	no	unibas	ludwig	1	0	sig
readall	unibas	yes	yes	no	no	no	uzh	obj.5	no	uzh	ute	1	0	sig

Table 2: The Privilege table

own interest that only authorized peers can read their mapping policies. Moreover, changes to the mapping policies are always reviewed by both contractual peers. Since the protection of the privileges in the *DA* is weaker, the directory can contain incorrect entries which makes authorization checking more challenging.

In summary, the *DA* is more suitable for dynamic environments because the obstacle to join the network for the first time is much lower than in the *MA*. Information gathering, and thus the creation of authorizations between peers, is easier. This makes the *DA* the preferred choice for applications with changing participants that are open to new members (e.g., simple data exchange applications). Conversely, the *MA* is the appropriate solution for relatively static collaborations with high security demands (e.g., a PDMS with medical data). The initial effort to participate in the network is higher but the granted privileges are better protected.

9. CONCLUSIONS AND OUTLOOK

We presented a novel approach to establish a P2P based access control mechanism in a collaborative way. Through the use of XACML policies we provide an application independent, powerful and flexible global access control model. We presented two possible approaches how this model can be distributed in a network. We currently implement a P2P client that supports both approaches, for later evaluation and testing.

10. REFERENCES

[1] K. Berket et al. PKI-Based Security for Peer-to-Peer Information Sharing. In *P2P 2004*, pages 45–52, 2004.

[2] O. A. C. T. Committee. Core and Hierachical Role Based Access Control (RBAC) Profile of XACML v2.0, 2005. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf`.

[3] O. A. C. T. Committee. eXtensible Access Control Markup Language (XACML) Version 2.0, 2005. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.

[4] O. A. C. T. Committee. XML Digital Signature profile of XACML v2.0, 2005. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-dsig-profile-spec-os.pdf`.

[5] B. Crispo et al. P-Hera: Scalable Fine-grained Access Control for P2P Infrastructures. In *ICPADS'05*, pages 585–591, 2005.

[6] J. F. da Silva et al. Policy-Based Access Control in Peer-to-Peer Grid Systems. In *Grids 2005*, pages 107–113, November 2005.

[7] S. De Capitani di Vimercati and P. Samarati. Authorization Specification and Enforcement in Federated Database Systems. *Journal of Computer Security*, 5(2):155–188, 1997.

[8] J. R. Douceur. The Sybil Attack. In *IPTPS '02*, pages 251–260, 2002.

[9] M. Franklin et al. From Databases to Dataspaces: a new Abstraction for Information Management. *SIGMOD Rec.*, 34(4):27–33, 2005.

[10] M. Harren et al. Complex Queries in DHT-based Peer-to-Peer Networks. In *IPTPS '02*, pages 242–259, 2002.

[11] D. Heimbigner and D. McLeod. A Federated Architecture for Information Management. *ACM TOIS*, 3(3):253–278, 1985.

[12] D. Jonscher and K. R. Dittrich. An Approach for Building Secure Database Federations. In *VLDB 94*, pages 24–35, 1994.

[13] G. Miklau and D. Suciu. Controlling Access to Published Data Using Cryptography. In *VLDB '03*, pages 898–909, September 2003.

[14] NIST. Role Based Access Control. ANSI INCITS 359-2004, February 2004.

[15] T. O'Reilly. What is Web 2.0. Design Patterns and Business Models for the Next Generation of Software. `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, 2005.

[16] E. Rissanen et al. XACML v3.0 Administrative Policy Version 1.0 Working Draft 15. `http://www.oasis-open.org/apps/group_public/download.php/21682/xacml-3.0-administration-v1-wd-15.zip`, 2007.

[17] E. Rissanen and B. S. Firozabadi. Administrative Delegation in XACML - Position Paper. In *W3C Workshop on Constraints and Capabilities for Web Services*. `http://www.w3.org/2004/08/ws-cc/erbsf-20040902`, 2004.

[18] R. Sandhu and X. Zhang. Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In *SACMAT '05*, pages 147–158, 2005.

[19] I. Stoica et al. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In *SIGCOMM 2001*, pages 149–160, 2001.

[20] C. Sturm. Orchestrating Access Control in Peer Data Management Systems. In *EDBT Workshops*, pages 66–74, 2006.

[21] M. R. Thompson et al. Certificate-based Authorization Policy in a PKI Environment. *ACM Trans. Inf. Syst. Secur.*, 6(4):566–588, 2003.

[22] H. Tran et al. A Trust based Access Control Framework for P2P File-Sharing Systems. In *HICSS'05*, page 302c, 2005.

[23] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In *ISSS '02*, pages 42–57, 2002.