Collaborative group membership and access control for JXTA

Joan Arnedo-Moreno and Jordi Herrera-Joancomartí
Estudis d'Informàtica, Multimèdia i Telecomunicacions
Universitat Oberta de Catalunya
Rb. Poble nou 156, 08018 Barcelona
{jarnedo,jordiherrera}@uoc.edu

***Abstract*— This paper presents a proposal for group membership and access control services for JXTA, both based on the principle of self-organization and collaboration of peer group members. The need for collaboration strengthens the resistance against free riding and eases management of revocation data. The proposal prioritizes group autonomy and makes use of the concepts of web of trust and small world phenomenon in order to achieve its ends, distancing itself from approaches based on centralized PKI models or trusted third parties external to the group. It also offers an alternative to the basic group membership services distributed with the JXTA platform implementations.**

Keywords: Access control, peer-to-peer, security, group membership, JXTA, distributed systems, web of trust, Java.

I. Introduction

In a peer-to-peer environment, all involved parties must collaborate in order to provide basic services, such as content or messaging. It is also assumed[1] that all peers have equivalent capabilities, and then the existence of a central server with more processing power is no longer necessary. Usually, peer-to-peer environments are conceptualized as a global overlay network without any kind of logical segmentation or segregation as far as resource availability is concerned.

JXTA [2] introduces the concept of *peer group*: a collection of peers with a common set of interests. This concept is one of the main foundations of the JXTA architecture and is prevalent throughout all its specification[3]. Offering the possibility to create different (but not necessarily disjoint) groups of peers operating under the same overlay network allows to segment the network and offers a context to peers for publishing and accessing different services.

Peer group boundaries provide a secure framework in order to grant or deny access to the offered services. Peer groups form logical regions whose boundaries limit access to group resources, in a way similar to a VPN [4], but operating at the application layer. Other interesting uses are the ability to provide a scoping or monitoring environment, where different classes of traffic and advertisements are limited to peer group members. In order to enable peer group security, JXTA defines the basic primitives for group membership and access control.

This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

In this paper, a specification for group access control using the core JXTA services is presented. The idiosyncracies of the JXTA platform are taken into account in order to create a system which is fully compliant with its specification.

This paper is organized as follows. Section II presents an overview of the generic JXTA group access control framework, formed by the Membership and Access Services, describing the current proposals for such services. In section III, the base model for group access control used for our specification is briefly introduced. Section IV describes the group membership specification details, explaining how the Membership and Access Services are deployed using the JXTA framework and which are the required support services. Section V concludes the paper and gives some guidelines for further research.

II. An overview of group membership in JXTA

In order to achieve full control of group membership, JXTA uses the Membership and the Access Services. Both are core services which make use of the base JXTA protocol specification in order to achieve their ends. The Membership Service manages identities within a peer group, providing each group member a *credential*. Peers may include this credential in messages exchanged within a group in order for each other member to know who is making a request. With this information, the JXTA Access Service evaluates the credential when a service is accessed and decides whether the request will be granted or denied.

A. Membership Service

The Membership Service allows every peer to establish its identity within a group and obtain a credential, it must successfully *join* it. Before a peer may join a group, it must be authenticated by providing a correct *Authenticator* to the Membership Service. An Authenticator contains all the required information in order for the Membership Service to check that the requested identity can be granted. Each different Membership Service specification provides its own definition of an Authenticator, suited to its needs and inner workings.

The join process is divided in three distinct steps: setup, application and validation, as depicted in Figure 1.

In the setup step, the peer chooses which authentication method will be used for the whole process. This is achieved by choosing an *AuthenticationCredential* and sending it to the

Membership Service via the *apply()* method. Some examples of specific authentication methods are interactive authentication (via a GUI), plain text authentication (programatical), etc. The *AuthenticatorCredential* also serves as a way to identify and allow interaction with the peer even though it is not part of the group yet. If all parameters are correct and the choice is feasible, the peer receives an Authenticator from the Membership Service.

In the application step, the peer completes the Authenticator with all necessary information and tests its correctness with the *isReadyforJoin()* method. It will not be possible to join the group until the Authenticator is correctly initialized.

Finally, in the validation step, joining the group is possible if the Authenticator is correct. The Membership Service checks whether the peer may assume the claimed identity and creates a credential. Since the join method is provided by each implementation of the Membership Service, the identity ownership check may be as complex as necessary.

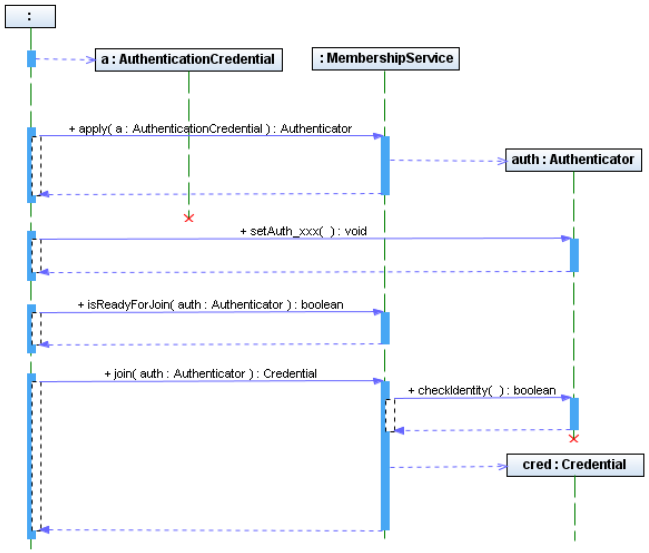

Fig. 1. Peer group join process

In case that a peer decides to give up membership to a specific group, a *resign* method exists. When this happens, the credential generated in the join process is discarded. Group resignation is voluntary, the Membership Service does not support active membership revocation triggered by other members.

B. Access Service

The credentials generated in the join process may be sent whenever a group service is accessed, as part of the protocol exchanges. The JXTA Access Service provides mechanisms in order to check them, allowing services to decide whether access should be granted or not.

The sequence diagram for using the Access Service is shown in figure 2. A single operation is offered to the peer group in order to check a credential for a privileged operation.

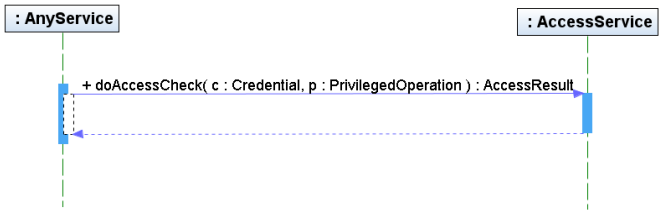

Fig. 2. Access control check via Access Service

The possible results are DISALLOWED, PERMITTED, PERMITTED_EXPIRED (the operation would be permitted but the credential has expired) and UNDETERMINED.

It is very important to remark that the Access Service is fundamental in order deploy any group access control model under JXTA. The reason is that JXTA platform allows a peer to directly instantiate any group and access its published services without credentials (using a ”nobody”-like default identity). The use of credentials is optional. As a result, unless the use of credentials is enforced via the Access Service, the process of joining a group via the Membership Service and obtaining a proper credential becomes pointless.

C. Existing service specifications

The JXTA reference implementation, as far as version 2.5[5], provides three Membership Services.

The *None* Membership Service is intended for peer groups which need no authentication. Since any peer may claim any identity, it is recommended that credentials should only be used within the group for purely informational purposes.

The *Passwd* Membership Service relies on a Unix-like username and password pair for peer authentication. In order to claim an identity, the correct password must be provided. The list of pairs (username and password) is distributed to all group members, which means that the password file equivalent roams freely through the overlay network. This group membership service was created as a sample and a means of testing, since it is completely insecure. For that reason, it is advised in the JXTA documentation that it should never be used in any serious application.

The default Membership Service is PSE, which stands for Personal Security Environment. This service provides credentials based on PKIX[6] certificates. However, it must be taken into account that the Authenticator for this service exclusively relies on passwords: the peer is authenticated to the group by being able to access the keystore which holds its private key. This means that the join method itself is not based on digital signatures or certificates, just the credential format itself. In fact, as presented, PSE is more of a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates, since it provides no clear structure of how trust is managed in a peer group (whose signatures are trusted and which peers are allowed to sign certificates).

It is interesting to point out the implications of the fact that under the PSE Membership Service peers are authenticated only by being able to access a keystore. The Membership

Service is not concerned with the validity (signed by a proper Certification Authority or not revoked) of certificate chains. As a result, anybody with access to a private key and a certificate (a self-signed one is sufficient) will be able to join a group using PSE. Those peers which hold the necessary information in order to generate a correct PSE Authenticator, but are not really group members because their certificate is not properly signed are named *interlopers*. These peers can become an annoyance, but are easily dealt with.

Apart from those in the basic implementation of JXTA, other Membership Service proposals exist, such as the one presented in [7], based on a centralized PKI and a basic challenge-response [8] protocol as a means for authentication, instead of passwords. This proposal also goes beyond the basic group membership services, providing a method which peers may use in order to authenticate the group itself in the join process. However, this implementation heavily relies on both an external centralized LDAP server and a sign server. The LDAP server provides peer certificate management and the sign server deals with group authentication. In order to do this, it keeps a secure copy of a private key which represents the whole group. The existence of those external entities goes completely against the basic principles of peer-to-peer and makes the system unfeasible in ad hoc environments. Furthermore, the proposal does not specify who configures or manages both central servers. It is assumed that some group administrator will do it out-of-band, which moves the proposal away from a pure peer-to-peer model.

Another proposal can be found in [9], but it is very similar to the passwd Membership Service, sharing most of its pros and cons.

As far as the Access Service is concerned, the current JXTA reference implementation offers three kinds of access control.

The *Always* Access Service, which does not really check for access control and allow any operation. It is the default Access Service for peer groups.

The *simpleACL* Access Service uses Access Control Lists in order to establish which identities may perform the different group operations. The access lists are distributed as parameters within the peer group advertisement.

The *PSE* Access Service provides an interface to PKIX certificate path validation. A trust anchor is set for the validation process and all credentials are validated against this anchor in order to decide whether the the operation is permitted or not.

It is very interesting to point out that all currently provided approaches to the Access Service are strictly tied to peer identity (operations are evaluated according to the claimed identity) and do not check group membership itself. Membership is assumed, even though it may not be always the case, as previously exposed.

III. Collaborative peer group access control

A proposal for group access control in peer-to-peer environments is presented in [10]. This proposal takes into account the nature of peer-to-peer networks, being fully decentralized and paying special attention to the autonomy of its members and self-organization. All necessary services are provided by its members and are based upon their collaboration, avoiding dependency from external group entities. For that very reason, this proposal may be specially suited for JXTA group membership and access control services.

This model uses the concept of web of trust defined in PGP[11], but in this case trust relationships are used as proof of group membership, instead of establishing an identity. Whenever peer A creates a trust relationship with peer B, it is vouching for B's group membership to other group members. It must be pointed out that this model is specifically concerned in providing proof of peer group membership, not peer identity.

Two different sets of peers exist within a peer group: those which are allowed to register new members, named *patron peers*, and those who are not. No initial assumption is made regarding the cardinality of both sets, how a peer becomes part of each set or which are the restrictions in order to do so. A peer may decide to switch sets at any time.

Since two different sets of peers exist, two different types of trust relationships are distinguished between peers, depending on which set both peers belong.

Patron relationships are established from peers which may register to peers which do not. These trust relationships are unidirectional, like web of trust relationships. When peer A signs peer B's key, it will be considered to be its patron within the group.

Backbone relationships are established between peers which both may register, and are considered to be bidirectional. In order to create, backbone relationships, both peers always sign each other's public key.

Both types of trust relationships are created in the same way as in a standard web of trust, by signing the trusted peer public key, generating a certificate with a specific date of expiration. This certificate specifies which kind of trust relationship it is for, reinforcing the fact that a peer is member of the group. Trust relationships are bound to a specific peer group, they are not usable across different groups.

Since peers are autonomous, each one exclusively manages the information concerned about itself. This means that each peer will have information related to:

- Its own private/public key pair.
- The signatures from other peers.
- The list of peers trusted by him.

To access any other information, other peers must be asked to provide it.

In order for a new peer, B, to register to the group, it must apply for membership to any patron peer, such as A. If agreement is reached, a patron relationship is established and A becomes B's patron for this group. The model does not impose any restriction on deciding why a new peer is accepted into the group. This decision will be up to the group policies or the individual decision of A, and goes beyond the scope of the model.

Under this model, in order for a peer C to provide proof of membership, a trust path must exist between the peer which must grant access and C. Both kinds of trust relationships are eligible when searching this path. However, this trust path

must accomplish an additional condition that all contained signatures must be bound to the peer group C is trying to access. In figure 3 a case is shown where A may be able to correctly recognize C as a group member.

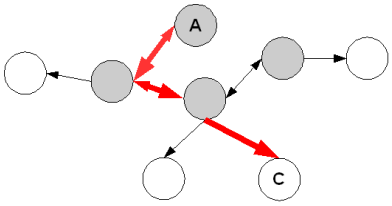

Fig. 3. Finding a trust path between peers A and C

In order to minimize the length of trust paths chains and provide better redundancy, when A successfully authenticates C, A checks whether it shares a backbone relationship with C's patron. If it does not exist, A may ask C's patron to create a backbone relationship between both of them. As the group life progresses, connection between patron peers will provide a better operation framework.

The model also accommodates to the possibility that n peers may become patrons of B. Under this assumption, such relationships may be created once B is already part of the group during group registration. Any peer may ask for more patrons at any time. This assumption enables the group to provide redundancy in order to solve possible availability problems, such as cases when authentication might fail because insufficient peers are online, a key issue in peer-to-peer and adhoc newtorks. Furthermore, this enables implementing stricter group policies or scenarios where a minimum number of patrons must be collected before B is considered to be part of the group.

IV. Group access control specification for JXTA

In order to adapt the model presented in the previous section to the JXTA specification, all its processes must be integrated into the core JXTA group membership and access control services: the Membership and Access Services. Additional support services are specified when necessary in order to encompass the full trust path creation and evaluation.

The summarized steps that a peer B must follow in order to access a service under this model follow:

1) B searches a patron peer for that group.
2) B requests to the patron its signature of B's public key.
3) If the patron agrees, a group credential which will act as proof of group membership is obtained.
4) (Optional) B is free to repeat steps 1-3 with additional patrons at any later stage.
5) B accesses a group service through peer A.
6) A asks for B's credential(s).
7) A tries to find a trust path from itself to B.
8) The path is evaluated. If correct, access based in group membership is granted.
9) (Optional) A new backbone relationship is created between A and B's patron

Our specification meshs with JXTA by using the Membership Service for steps 1-4, those concerned with credential generation, whereas the Access Service is to be concerned with steps 5-9, those concerned with service access.

A. Membership Service

A peer which wants to join a peer group must apply for membership via the Membership Service. Accessing the Membership Service, as well as all its available services, is based on the precondition that a peer has previously instantiated the group via a *group advertisement*. Group advertisements are sent by those peers which want to publicize the group (usually, at least, the group creator), and must be periodically updated, since it has a set lifetime. Once no more advertisements are available in the network, it will be impossible to instantiate a group.

Such advertisements contain all the required information parameters specific for its Membership Service implementation. For that reason, deploying a new Membership Service means defining the format of a group advertisement. For this Membership Service specification, a list of patron node ID's known by the peer which published the advertisement is included as a service parameter. This way, prospective members may know where to get a properly signed certificate in order to access group services. As the list of patron nodes changes, new advertisements may take it into account when published.

An example of peer group advertisement is shown in listing 1 (Note: ID's have been shortened).

XML Listing 1 - Peer group advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xml:space="preserve" xmlns:jxta="http://jxta.org">
<GID>urn:jxta:uuid-2AD...5F02</GID>
<MSID>urn:jxta:uuid-DEADBEEFDEAFBABAF...000000010306</MSID>
<Name>SampleGroup</Name>
<Desc>Collaborative Membership Service Group</Desc>
<Svc>
<MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000505</MCID>
<Parm type="jxta:PatronList">
<PeerGroup>urn:jxta:uuid-2AD...5F02</PeerGroup>
<Patrons>
<Patron>urn:jxta:uuid-596...4003</Patron>
<Patron>urn:jxta:uuid-596...1B03</Patron>
</Patrons>
</Parm>
</Svc>
</jxta:PGA>
```

In our Membership Service specification, an interface named *CryptoManager* is defined in order to arbitrate the join process. This interface is passed as the authenticating parameter to the Authenticator and is used in the application and validation steps, authenticating the peer using a challenge-response protocol invoked via a callback mechanism.

The main reason for the CryptoManager is acting as a proxy for any cryptographic provider. It enables the integration of peer group authentication with any kind of cryptographic provider such as hardware cryptographic tokens. This approach takes into account the fact that not all cryptographic providers are accessed via plain text passwords (for example,

using biometrics). Even in cases were passwords are used, sometimes, each provider has its methods for accessing private keys (for example, a special GUI integrated into the operating system in the the Windows CryptoAPI[12]). Applications will implement the CryptoManager interface according to the needs of the specific cryptographic provider being used. Decoupling cryptographic providers from the Membership Service also allows to manage credentials via out-of-band methods, instead of necessarily relying on the application.

This is in contrast with PSE, which by default uses passwords in order to access a java keystore. This keystore may be stored in the peer local cache or as a file accessed via a URL. In the former case, its management is encapsulated in a transparent way via the JXTA Cache Manager. Unfortunately, in this case, it is not possible to easily manage certificates via out-of-band methods. In the latter case, it is not possible to use any other kind of cryptographic provider which does not rely on files protected by a plain text password.

The basic methods of CryptoManager are:

- *initialize*: Sets up the cryptographic provider, in order to be correctly accessed later. Usually, complex providers need some initialization. If not really necessary, this method may be left blank. If additional parameters are needed to use the provider, it is possible to define additional methods which may be called before this one.
- *check*: Checks whether initialization has been successfully completed and it will be really possible to access stored keys and certificates.
- *getCerts*: Obtains the list of stored certificates in the cryptographic token. The Authenticator processes this list in order to assemble only those certificates related to the peer group to be joined.
- *sign*: Signs data. During this method call, the cryptographic provider prompts for the necessary information in order top access the key.

A sequence diagram summarizing the full process is shown in figure 4. The full join process is not shown in this figure, but only the part which is specific for our Membership Service specification: from the moment an Authenticator has been made available up to the final step. Once the join is successfully invoked, a group credential is generated.

Our Membership Service uses a list of certificates as a credential. Even though open to different types of certificates, as will be further explained in subsection IV-A.2, in the current specification, X.509 certificates are used. X.509 certificates have been chosen because they are the most widely used in cryptographic providers. The peer group ID is stored in the O certificate field, whereas peer ID is stored between the OU and CN fields (so peer $ID = OU + CN$), because of space restrictions in the X.509 format. The type of relationship (backbone/patron) is stored on the Location Field.

A sample credential is shown in listing 2 (Note: X.509 Certificates have been shortened).

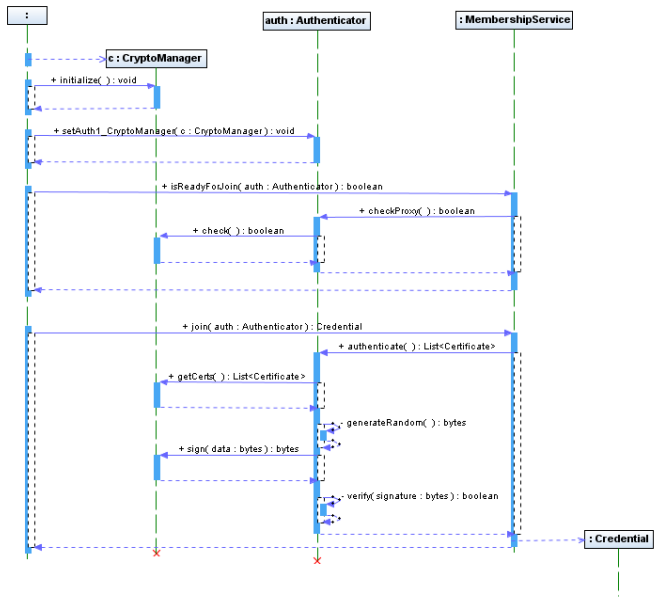

Fig. 4. Peer group join process using CryptoManager

XML Listing 2 - Peer Credential

```
<!DOCTYPE jxta:CollabCred>
<jxta:CollabCred algorithm="SHA1withRSA"
type="jxta:CollabCred" xml:space="preserve"
xmlns:jxta="http://jxta.org">
<X509Certificate>MIIC...BIBUAMw==</X509Certificate>
<X509Certificate>MIID...8dLvZ96==</X509Certificate>
</jxta:CollabCred>
```

Even though a credential has been established, the CryptoManager will still operate within the Membership Service as an interface to the cryptographic provider until the peer resigns from the group.

It is possible that no proper group credentials are available (basically, when a new member tries to join the group). In that case, the credential is nevertheless generated, but such peer is only able to access a very specific set of group services: the Patron Discovery and Sign Services. These services serve as methods in order to provide peers with proper credentials. They are also a special case, since they do not check group membership and are available to any peer which is able to instantiate the group, as it will be made obvious.

Once a proper credential is created, the peer will reauthenticate, so a proper credential is eventually generated by the Membership Service.

1) Patron Discovery Service: This service retrieves a list of peer group patrons. Even though the peer group advertisement includes a list of patrons, it enables updating the patron list for someone who has already joined the peer group without reinstantiating it. This service also takes into account the fact that some patrons may go offline.

Any group member may run this service. Non-patron peers response with a list obtained from its own stored patron relationships. Patron peers response a list obtained from its acknowledged backbone relationships.

The format of the patron discovery service query and response is shown in listings 3 and 4 (Note: ID's have been shortened).

XML Listing 3 - Patron Discovery query

```
<collab:PatronDiscoveryQuery>
<PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
<Threshold>4</Threshold>
</collab:PatronDiscoveryQuery>
```

XML Listing 4 - Patron Discovery response

```
<collab:PatronDiscoveryResponse>
<PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
<Patrons>
<Patron>urn:jxta:uuid-596...CB03</Patron>
<Patron>urn:jxta:uuid-596...DB03</Patron>
<Patron>urn:jxta:uuid-596...6003</Patron>
<Patron>urn:jxta:uuid-596...EE03</Patron>
</Patrons>
</collab:PatronDiscoveryResponse>
```

The fields used in the messages are the following ones.

- *PeerGroupID* - The unique identifier of the peer group whose patrons are being discovered. This identifier is included in the response in order to allow proactive responses, those sent without a previous query, to be correctly processed.
- *Threshold* - A number specifying the maximum number of patrons that should be sent by the responding peer.
- *Patrons* - An element containing a list of patron peer identifiers. The identifiers are relative to the peer group being asked.

It should be noted that the response has the same format as the service parameter included in the peer group advertisement shown in section IV-A.

2) Sign Service: In order for a prospecting member to become a fully fledged group member, a proper credential must be obtained by getting a certificate signed by a patron peer. The Sign Service allows the request of such signatures without the need of out-of-band methods. This service supports both the creation of both patron and backbone relationships. By using this service, it is also possible to obtain credentials from several patrons.

The moment a patron must decide whether to accept or not a query is a very important step. The outcome of such decision is left up to each specific application, since it is impossible to develop a policy that may apply to any scenario. Acceptance will generally be based on the fulfillment of some requirements before joining the group (such as committed resources). An interesting possibility is using trust evaluation schemes [13], [14]. Another acceptable possibility is just sending the request to the upper level application and relying on direct user input. The user will decide according to some out-of-band information, such as in an invitation based system.

Registering the signing service in a non-patron peer is pointless, since the certificate will not be a valid one. Even though this behaviour may be potentially disrupting for unsuspecting peers, it is easy to identify which rogue peers are signing requests by comparing patron lists retrieved from different peers, or as soon as a service access is denied.

The details of the Sign Service are shown in the query and response formats in listings 5 and 6.

XML Listing 5 - Sign Service query

```
<collab:SignQuery>
<PeerID>urn:jxta:uuid-596...CB03</PeerID>
<PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
<Data type="pkcs#10">MIIB...PYNvDvHf1</Data>
<Commendation>...</Commendation>
</collab:SignQuery>
```

XML Listing 6 - Sign Service response

```
<collab:SignResponse>
<Status>OK</Status>
<PeerID>urn:jxta:uuid-596...CB03</PeerID>
<PeerGroupID>urn:jxta:uuid-282...8C02</PeerGroupID>
<Data type="x509certificate">MIIC...BIBUAMw==</Data>
</collab:SignResponse>
```

The sign query and response parameters are listed here.

- *Status* - Sign request result. The request may have been granted (OK) or denied (DENIED). The responding peer may also respond that the query has been correctly received but it is still to be processed (PENDING). This status is specially useful for application which rely on direct user input in order to accept or deny a sign request.
- *PeerID* - The unique identifier of the peer who is asking for a signature.
- *PeerGroupID* - The unique identifier of the peer group for which a credential is being requested.
- *Data* - The request/credential raw data (base64 encoded in case of binary data). This element will only exist if a sign request has been accepted (the Status field is OK). It also contains a *type* attribute, which denotes a special identifier specifying the type of request (for a sign request) or credential (for a sign response). In the case of the sign response, the identifiers chosen for the different types of sign responses are those defined in xmldsig[15]. This will allow an easy integration with XML signature. This is a desirable feature since all JXTA protocols are based on XML. In our current specification, PKCS#10 and X.509 certificates are used.
- *Commendation* - Additional optional information that a peer may provide in order to get the sign request accepted. Since the acceptance of a sign request is solved at application level, this field may contain any kind of data. The specific data format will be defined by each individual application, according to its own needs. The query message just acts as a transparent envelope. Nevertheless, an existing valid trust path may be used as a commendation when creating redundancy and shortening long paths, as explained in section III.

Both the Patron Discovery and Sign Services are not encapsulated into the Membership Service. Since the model allows the creation of multiple trust relationships during standard group operation (steps 4 and 9 in the summary list at the beginning of this section). Then, it must be possible to create

them once the group has been joined without the need of rejoining via the Membership Service.

B. Access Service

As exposed in subsection II-B, proof of peer group membership must be required when accessing remote resources. This is done via de Access Service. It should not be done during the join process, since group instantation and the join process are locally executed. That means that it would be trivial for a rogue peer to modify the local code in order to always successfully join the group. In our proposal, this may still happen, but it will be pointless since proper credentials cannot be generated only using local data, they are generated through collaboration with other peers. And unless a proper credential is somehow generated, no access will be granted at a later stage.

Since the group membership and access control model is based on collaboration between peers, the Access Service must operate in a fully asynchronous way in order to take into account the nature of adhoc environments. Some delay is to be expected between access to the service being asked and a final decision regarding access control being reached. For that reason, in our specification the doAccessCheck operation will return UNDEFINED until it is ready to provide a definite answer. The final behaviour is similar to that of the isReadyForJoin in the join process (see section II-A), which must be periodically polled in order to know when it is possible to proceed.

In order to retrieve and evaluate a trust path under this premises, the doAccessCheck operation defined by the JXTA Access Service uses two additional auxiliary services: the Trust Path Discovery Service and the Credential Retrieval Service.

1) Trust Path Discovery Service: The Trust Path Discovery Service provides a mechanism to learn whether a trust path between two peers exists, and through which peers this trusted path passes. This service correctness requires collaboration between different peers in the group, since the group membership model is based on the principle that each peer is autonomous and only holds data mostly related to itself. It also assumes that no peer maintains memory or state regarding processed queries and the whole process acts in a fully asynchronous way.

In order to retrieve the trust path between two peers, the source peer propagates queries across all its trusted peers. This query contains the final peer ID in the trust path and its own ID as the source peer. In the case that peers receiving the query are not able to retrieve a trust path to the final peer by themselves, they propagate the same query to its own trusted peers, but including its own ID as an intermediate peer. As this query reaches new peers, the query grows to collect the trust path, adding its own ID at the end of the current list, until a peer which knows a trusted path to the final destination is reached. That peer sends the final answer directly to the source peer. A summary of this whole process is depicted in figure 5.

Loops are avoided by comparing the currently retrieved trust path in the query with the list of trusted peers in the processing

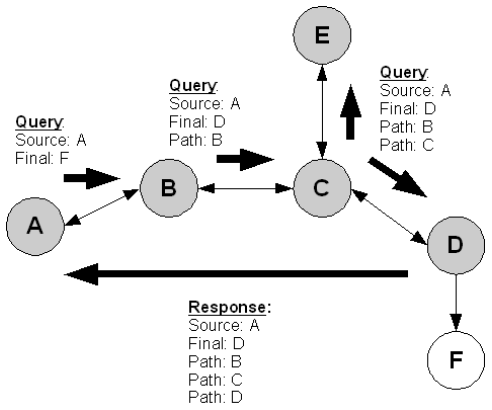

Fig. 5. Trust path discovery query propagation

peer. If a peer already appears in the list, the query is not propagated across that trust relationship.

It must be pointed out that this service may generate several responses to the source peers, since several trust paths may exist. It must also be noted that the trust path does not need to be retrieved one peer at a time. If a peer already knows a path to the final peer, even though there is no direct trust relationship (for example, because of cached responses to its own queries), it may still respond the query constructing the full trust path. In that case, the peer stops propagating queries.

The Trust Path Discovery query format is detailed in listing 7. The contents of the response are the same.

XML Listing 7 - Trust Path Discovery query/response

```
<collab:TrustPathQuery>
<SourcePeerID>urn:jxta:uuid-596...CB03</SourcePeerID>
<FinalPeerID>urn:jxta:uuid-596...6003</FinalPeerID>
<PathPeerID>
<Position>1</Position>
<PeerID>urn:jxta:uuid-596...DB03</PeerID>
</PathPeerID>
</collab:TrustPathQuery>
```

The query and response parameters follow.

- *SourcePeerID* - Identifier of the starting peer in the trust path.
- *FinalPeerID* - Identifier of the final peer in the trust path.
- *PathPeerID* - Identifier of an intermediate peer in the trust path. These fields are ordered according to transit from the source peer to the final peer.

Since each peer manages its own list of trusted peers, the trust discovery path also allows to check revocation status, since no queries will be propagated through revoked trust memberships. This is in contrast with other membership services such as PSE, where revocation does not exist. It is only possible to voluntarily resign from a group.

2) Credential Retrieval Service: The Credential Retrieval Service provides a method for peers in the group in order to retrieve credentials from other peers, as well as asking for its current revocation status. Once a trust path is located, all credentials may be retrieved using this service in order to

proceed to its validation. Several retrieval petitions may be encapsulated in a single request, in order to improve efficiency. Again, responses may be sent in a proactive way without the need of a previous query.

All requests are only relative to the information stored within a peer (see section III).

The Credential Retrieval Service query and response format are shown in listings 8 and 9.

XML Listing 8 - Credential Status query

```
<collab:CredRetrievalQuery>
<PatronPeerID>urn:jxta:uuid-596...CB03</PatronPeerID>
<TrustedPeerID>urn:jxta:uuid-596...6003</TrustedPeerID>
/<collab:CredRetrievalQuery>
```

XML Listing 9 - Credential Status response

```
<collab:CredRetrievalResponse>
<TrustedPeerID status="OK">urn:jxta:uuid-596...6003</TrustedPeerID>
<Data type="x509certificate">MIIC...BIBUAMw==</Data>
</collab:CredRetrievalResponse>
```

- *PatronPeerID* - The credential generated by the patron peer with this ID for the peer which receives the request is being asked. This field may appear multiple times, containing different ID's.
- *TrustedPeerID* - The credential status for the peer with this ID is being asked.
- *TrustedPeerID* and *status*- Response tied to a TrustedPeerID request field.Credential status for a peer with a specific ID. The status attribute may be OK, REVOKED or UNKNOWN. The latter case may occur if the query asks for the status of a credential which was not generated by the responding peer.
- *Type* and *Data* - This field is a response to a PatronPeerID request field. It accomplishes the same function as the ones used in the Sign Service, explained in subsection IV-A.2. The ID from the original request may be obtained form the credential content.

V. Conclusions and further work

A specification of peer group membership and access control for JXTA using a web of trust based model has been presented. This specification integrates with the core services provided by JXTA in order to control peer group membership: the Membership and Access Services. Group management is achieved via additional services that take into account the idiosyncracies of JXTA's messaging capabilities: the Patron Discovery, Sign, Trust Path Discovery and Credential Retrieval services. Furthermore, the specification seamlessly integrates with different cryptographic providers, through the CryptoManager, in order to attain key management.

A concern in the current implementation is how the sign service is able to be sure that the presented public key to be signed is associated to the claimed identity. Even though the fact that the response is directly sent to the claimed ID's peer, which means that the attacker will not necessarily receive it, it is not a completely secure procedure yet, since the certificate will still be sent through the overlay network. Right now, some out-of-band procedure is necessary (which may be fine in some applications, but inconvenient). This problem may be solved using self-certifying keys[16], but at this stage it is still under research.

In addition, the Trust Path Discovery Service is still open to optimization, since right now it is entirely based in flooding backbone relationships, which is not efficient. It must be noted that it is not a full flooding mechanism, since requests are only propagated to trusted peers (which may be limited in number and are not necessarily equal no neighbouring peers), but it could be improved in terms of efficiency. Further efforts in research will be focused on this topic.

References

[1] Andrew Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.

[2] Sun Microsystems, "Project JXTA", http://www.jxta.org.

[3] Sun Microsystems, "JXTA v2.0 protocols specification", https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[4] Huston G. Ferguson, P., "What is a vpn?", Tech. Rep., Cisco Systems, 1998.

[5] "Jxta 2.5 rc1", June 2007, http://download.java.net/jxta/build.

[6] CCITT, "The directory authentication framework. recommendation", 1988.

[7] Zielinski K. Kawulok L. and Jaeschke M., "Trusted group membership service for jxta", in *Computational Science - ICCS 2004*, 2004, Lecture Notes in Computer Science Volume 3038.

[8] "Rfc 1994", http://tools.ietf.org/html/rfc1994.

[9] Lei Zhuang Zupeng Li, Yuguo Dong and Jianhua Huang, "Implementation of secure peer group in peer-to-peer network", in *Communication Technology Proceedings, 2003. ICCT 2003*, 2003, pp. 192–195.

[10] Joan Arnedo-Moreno and Jordi Herrera-Joancomartí, "Providing collaborative mechanism for peer group access control", in *Proceedings of the Workshop on Trusted Collaboration*. 2006, IEEEPress.

[11] S. Garfinkel, *Pgp: Pretty good privacy.*, O'Reilly and Associates Inc., 1994.

[12] Microsoft, "Msdn library. cryptography", 2007.

[13] Yeager W. Chen R., "Poblano: a distributed trust model for peer-to-peer networks", Tech. Rep., Sun Microsystems, 2001.

[14] T. Beth, M. Borcherding, and B. Klein, "Valuation of trust in open networks", *Proc. 3rd European Symposium on Research in Computer Security – ESORICS '94*, pp. 3–18, 1994.

[15] W3C, "Xml-signature syntax and processing", 2002.

[16] M. Girault, "Self-certified public keys.", *Advances in Cryptology - EUROCRYPT '91*, pp. 490–497, 1991.