Trusted Group Membership Service for JXTA

Lukasz Kawulok, Krzysztof Zielinski, and Michal Jaeschke

Department of Computer Science
AGH-University of Science and Technology
Krakow Poland

Abstract. This paper presents Group Membership Service for JXTA extended with single or bi-directional authentication. The proposed solution exploits certificates and PKI (Public Key Infrastructure). All information about the system's users is kept in the LDAP. An attention has been also paid to the problem of a private key secure protection.

1 Introduction

The concept of a peer group is very important to all aspects of the JXTA platform. Peer groups provide a way of segmenting the network space into distinct communities of peers organized for specific purpose providing the context for operations performed by services.

The goal of this paper is to present Group Membership Service for JXTA extended with single or bi-directional authentication. The proposed service will be called trusted Group Membership Service. The concept of the service assumes that a new member of group should be authenticated by a group in case of one-sided authentication and the group should be authenticated by a joining member in case of two-sided authentication. The proposed solution, in contrast to two already existing [1][2] implementations of a group access authentication for JXTA using passwords or null authentication, exploits certificates and PKI (Public Key Infrastructure).

The structure of the paper is as follows. First in Section 2 the concept of peer group and Membership service in JXTA has been briefly described. In this context the Trusted Group Membership Service concept has been formulated. Next in Section 3 implementation details of the proposed service are presented. The paper is ended with conclusions.

2 Trusted Groups versus Membership Service (Concept)

The first thing that the JXTA platform requires when bootstrapping is a World Peer Group, which is a peer group identified by a special Peer Group ID. The World Peer Group defines the endpoint protocol implementations supported by the peer, the World Peer Group itself can't be used to perform P2P networking. The World Peer Group is basically a template that can be used to either discover or generate a Net Peer Group instance. The Net Peer Group is a common peer group to peers in the network that allows all peers to communicate with each other.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 218–225, 2004.

Before a peer can interact with the group, it needs to join the peer group, a process that allows the peer to establish its identity within the peer group. This process allows peer groups to permit only authorized peers to join and interact with the peer group.

Each peer group has a membership policy that governs who can join the peer group. When a peer instantiates a peer group, the peer group's membership policy establishes a temporary identity for the peer. This temporary identity exists for the sole purpose of allowing the peer to establish its identity by interacting with the membership policy. This interaction can involve the exchange of login information, exchange of public keys, or any other mechanism that a peer group's membership implementation uses to establish a peer's identity. After a peer has successfully established its identity within the peer group, the membership policy provides the user with credentials. The membership policy for a peer group is implemented as the Membership service. In addition to the MembershipService class and its implementations, the reference implementation defines a Credential interface and an implementation called AuthenticationCredential. These classes, defined in net.jxta.credential, are used in conjunction with the MembershipService class to represent an identity and the access level associated with that identity [3]. Two steps are involved in establishing an identity within a peer group using the peer group's MembershipService instance:

1. **Applying for membership**. This involves calling the peer group's MembershipService's apply method. The apply method takes an AuthenticationCredential argument, which specifies the authentication method and desired identity. The method returns an Authenticator implementation that the caller can use to authenticate with the peer group.
2. **Joining the group**. The peer must provide the Authenticator implementation with the information that it requires to authenticate. When the peer has completed authentication using the Authenticator, the Authenticator's isReadyForJoin method returns true. The peer now calls the MembershipService's join method, providing the Authenticator as an argument. The join method returns a Credential object that the peer can use to prove its identity to peer group services.

When applying for membership, the peer making the request must know the implementation of Authenticator to interact with it. This is required because the Authenticator interface has no mechanism for the peer to interact with it. Using only the Authenticator interface, a peer can only determine whether it has completed the authentication process successfully and then proceed with joining the peer group.
The current implementations of Membership Service aren't especially useful. To provide proper authentication, a developer must develop a Membership Service of his own to manage the creation and validation of authentication credentials. In addition, the developer must provide a mechanism in his service to use the credentials and validate the credentials passed by other peers in requests to the service.

Although the Protocol Specification [3] outlines the concept of an Access service whose responsibility it is to verify credentials passed with requests, no implementation is provided in the reference JXTA implementation. The Resolver service's Resolver Query and Response Messages support a Credential element, but the contents of the element are never verified. For now, it appears that it is the responsibility of a developer to define his own Access service implementation and use it to verify credentials passed to his custom peer group services. As such, a developer

currently needs only to instantiate a peer group and can skip the steps of applying for membership and joining the peer group.

The proposed Trusted Membership Service concept assumes that Peer Advertisement issued by peer wanted to create trusted group contains additionally to standard information, already described, data specifying: localization of LDAP server, where certificates are stored; localization of Sign Server needed in case of bi-directional authentication; subject of group certificate.

This information is used in the process of joining trusted group. In case of the one-sided authentication the Trusted Membership Service activity follows the steps:

1. A peer invokes MembershipService's apply method of chosen peer group presenting its AuthenticationCredential in the form of Distinguished Name.
2. The peer group obtains certificate, containing its public key, corresponding to presented name from LDAP server and if the given peer has access permission to the group generates stochastic data that should be signed by the applying peer. This data are returned to peer by apply method.
3. The peer singes the data with its private key and returns back invoking the MembershipService's join method.
4. The peer group authenticate the peer using this data and its public key obtained early from certificates.

In case two-sided authentication the presented steps are extended as follows:

5. After successful peer authentication join method returns the peer group certificate.
6. The peer generates stochastic data which are signed using the peer group private key and sent back.
7. The authentication of the peer group is finally made using the peer group public key obtained early in the peer group certificate.

The proposed system architecture is depicted in Fig.1. The last problem which has to be solved is that P2P system concept doesn't not allow existing of any central point. A peer group could be created by every peer in any place. It means that in case of bi-directional authentication a private key has to be populated to each peer. It is rather risky solution from security point of view. In our service has been chosen the different solution. It exploits the concept of sign server which knows the group private key and signs data sent by a peer wanting to authenticate a peer group.

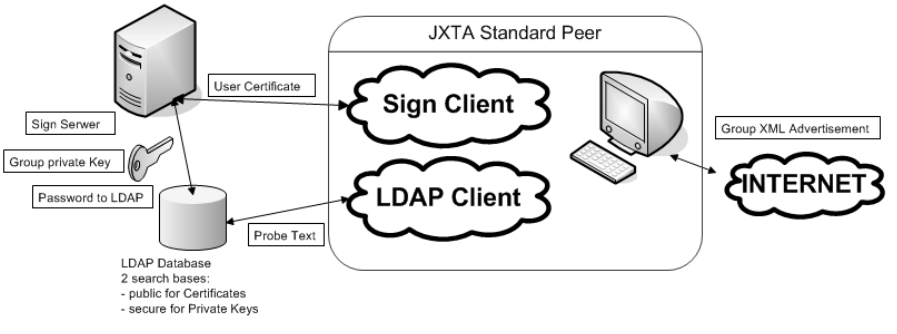

Fig. 1. Architecture of Trusted Membership Service implementation

3 JXTA Trusted Membership Service Implementation Details

In this section the details of JXTA trusted groups implementation are shortly described. We started from a new form of the trusted group advertisement specification and LDAP configuration description. Next activity of the Sign Server is specified. Finally the joining to the trusted group process is briefly described for one-sided and two-sided authentication case.

3.1 Trusted JXTA Group Advertisement

While creating a trusted group, a PeerGroupAdvertisment is created that contains additional information indispensable for proper operation of the membership service. Form of an example advertisement is:

```
<?xml version=„1.0”?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta=„http://jxta.org”>
<GID>urn:jxta:uuid-8A12B3AEC66E47BAB739999684D1705302</GID>
<MSID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE000000016FC5C1F3F
      7434F658A1C8C6CADFC43DB06</MSID>
<Name>VoteGroup</Name>
<Desc>Peer Group using Certificate Authentication</Desc>
<Svc>
  <MCID>urn:jxta:uuid-DEADBEEFDEAFBABAFEEDBABE0000000505</MCID>
    <Parm>
      <GroupDNName>EMAILADDRESS=CertGroup1@agh.edu.pl, zN=CertGroup
1, OU=ICS, O=AGH, L=Krakow, ST=Malopolska, C=PL
      </GroupDNName>
      <LdapServerPort>389</LdapServerPort>
      <LdapServerHost>192.168.2.4</LdapServerHost>
      <LdapServerSearchBase>dc=PP</LdapServerSearchBase>
      <SignServerPort>500</SignServerPort>
    </Parm>
  </Svc>
</jxta:PGA>
```

Fig. 2. Trusted group advertisment in XML format

Group advertisement contains information about the MembershipService. It is certified by the reserved UUID service number equal

DEADBEEFDEAFBABAFEEDBABE0000000505 in our implementation.

The similar parameters describing LDAP Server and Sign Server are specified in the properties file named mobisec.properties. They are used for the suitable servers configuration during JXTA system start up process.

3.2 LDAP Configuration Description

In order to keep the information about the system's users in the LDAP [4] database in a convenient way, we have defined two object classes which correspond to the user and group objects respectively. The classes contain the X509Certificate attribute we have created for keeping certificate in PEM or DER format [10].

Beside the X509Certificate attribute, the both classes possess attributes corresponding to all the elements of the distinguished name fields from the X.509 certificate, i.e. c, cn, mail, o, ou, st. Using these fields it is possible to search the given person. With the presented classes, we have created LDAP hierarchical database storing the group and user data. A similar structure beginning from the dc=PP root, which is used for storing private keys of individual groups is also constructed. For this purpose an additional attribute named PrivateKey and a new class for storing the elements discussed has been defined .

3.3 Sign Server

Sign Server is an independent application which performs a group authentication in case of the two-sided authentication. It functionality is to sign a text sample with a group private key. The activity of the Sign Server could described as follows – Fig. 3:

1. A dedicated thread reads the input from selected port (requests from peers);
2. After receiving a message, a dedicated thread for serving the request which contains the group certificate subject name and the text sample for signing is created.
3. The dedicated thread connects with the LDAP server to get the group private key.
4. Using this key the sample text is signed.
5. The signed text is sent back to the requesting peer.

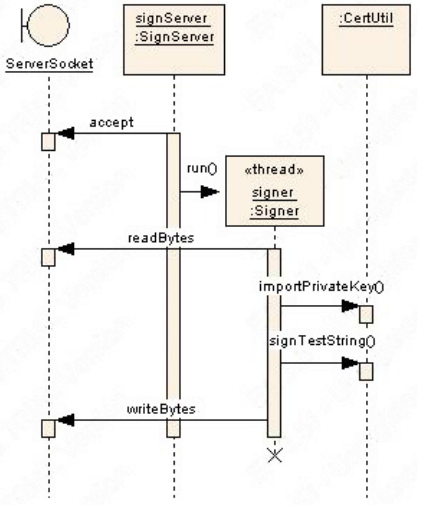

Fig. 3. Diagram of Sign Server operation sequence

There is a class named SignClient in the our utility package. After the correct initialization of the SignClient class, it may be used for communication with Sign Server. In the described project this class was used by the authenticator named CertAuthenticator in the middle of the group authenticating process.

3.4 One-Sided Authentication Sequence

This section presents how the authentication mechanism with certificates in the process of joining a group is used. Before the group starts operating, it must issue certificates to the authorized users and the group must possess itself a certificate signed by a trusted certification authority. In the simplified version, i.e. without authentication carried out by a group, the process is performed as follows [3]:

1. A group uses the CertMembershipService class to perform the authentication services. A participant who wants to join the group invokes an apply method of the CertMembershipService class, in order to establish an authentication method. This class allows only for authentication with use of the certificate defining the user's identity with his/her public key attached to it.
2. An apply method of the CertMembershipService class sends back to the user an object type CertAuthenticator (extension of the Authenticator interface), containing a group certificate and a random piece of data designed to be signed with a peer's private key. A participant fulfills a CertAuthenticator object with use of setAuthCertificate method of that object. This causes a peer's certificate (in the form of the object of the StructuredDocument class) and a piece of data sent by the group and signed with the peer's private key to be placed in the CertAuthenticator object.
3. A participant, in order to verify whether the CertAuthenticator object has been fulfilled properly, calls the method isReadyForJoin.
4. A participant gives a CertAuthenticator object containing its certificate and piece of the group data signed with a private key as a method parameter join of CertMembershipService class.
5. A group checks the certificate, verifying it by means of its public key (since the key was issued by the group) or with use of PKI and certificate issued by a trusted certification authority.
6. A group verifies the peer's identity, decoding a communicate with the peer's public key (attached to the certificate in the moment of authentication).

After the above sequence of operations is finished, a group is sure about the identity of the peer, which wants to join it.

3.5 Two-Sided Authentication Sequence

In the version with authentication carried out by a group (two side authentication) we have some additional elements in the presented above sequence. Above steps have to be repeated by the other side. It is peer who sent random piece of data designed to be signed with a group's private key. The group fulfills the CertAuthenticator object with use of setAuthCertificate method of that object. This causes a group's certificate and the piece of data sent by the peer and signed with the group's private key to be placed in the CertAuthenticator object and send back to the peer, and in the end it is the peer who decode the communicate with the group's public key and verifies the group's identity. In summary the following additional steps must be performed in the two-sided the authentication [3]:

7. While calling the CertAuthenticator object, defined in step one, the setAuth1GroupVerificationString method must be used in order to set random piece of information to be signed by a group.
8. In the very end the step verifying a group must be added. It requires the connection with the SignSerwer. The SignServer receives a sample text for signing. Then the Signserver takes a group private key from the LDAP database. Only the SignServer can make use of the additional database located on the LDAP server.

The sequence diagram for two-sided authentication is depicted in Fig.4. It presents the role of each component proposed by the trusted Group Membership Service for JXTA.

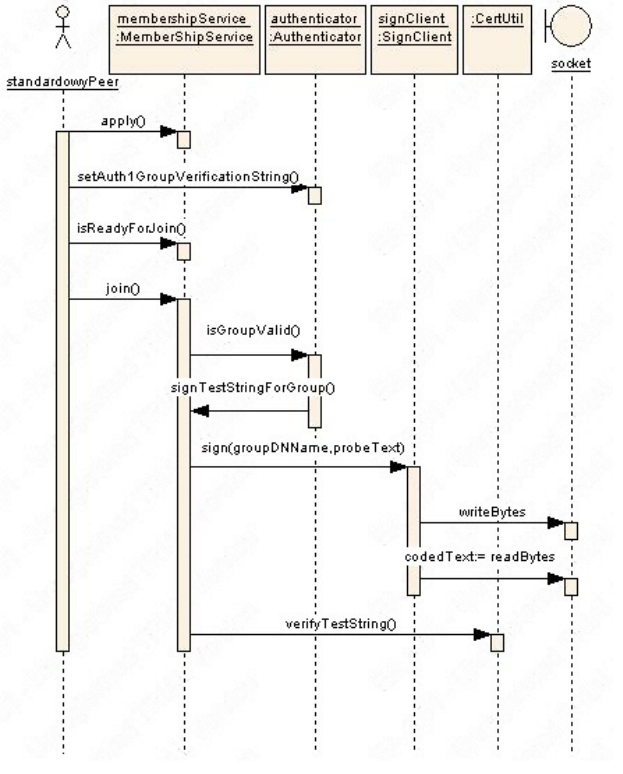

Fig. 4. Diagram of authenticator operation sequence in case of the two-sided authentication

4 Summary

The proposed implementation of Trusted Membership Service is fully compliant with JXTA reference implementation. Every effort has been put to make the system easy to use and to install. The service has been built as a separate package that can be used

whit standard JXTA packages. An attention has been also paid to the problem of a private key secure protection. This problem has not been discussed in more details as it rather a general issue. It is a problem for peers and for groups how to prevent the keys before stilling. For the peers we decided to encode the keys with a password – passphrase. For the group we have implemented a proxy object which has access to the key encoded with the password typed only once. The lifetime of the proxy object is limited for example by the user which is creating the group. This solution let us to avoid storing private key decoded on the disk. Now we are working on the proposed system practical verification by developing a range of applications. Existing related work is presented in [11], [12].

References

1. JXTA Project, http://www.jxta.org/
2. JXTA v2.0 Protocols Specification, http://spec.jxta.org/v1.0/docbook/JXTAProtocols.html
3. JXTA Book, http://www.brendonwilson.com/projects/jxta/
4. Iplanet LDAP description, http://www.cc.com.pl/prods/netscape/direct.html
5. Netscape Directory SDK 4.0 for Java Programmer's Guide, http://developer.netscape.com/docs/manuals/dirsdk/jsdk40/contents.htm
6. Security in Java Technology, http://java.sun.com/j2se/1.4.1/docs/guide/security/index.html
7. The Java Developers Almanac, http://javaalmanac.com/egs/javax.naming.directory/Filter.html
8. Certificates structure http://java.sun.com/j2se/1.4.1/docs/tooldocs/windows/keytool.html - Certificates
9. The PKI page, http://www.pki-page.org/
10. OpenSSL Project, http://www.openssl.org/
11. Security and Project JXTA, original paper January 2002 www.jxta.org/project/www/docs/SecurityJXTA.PDF
12. Poblano – A Distributed Trust Model for Peer-to-Peer Networks www.jxta.org/docs/trust.pdf