A survey on security in JXTA applications ⋆

Joan Arnedo-Moreno [a,*], Jordi Herrera-Joancomartí [b]

[a] *Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya, Rambla de Poblenou 156, 08018 Barcelona, Spain*

[b] *Escola Tècnica Superior d'Enginyeria, Universitat Autònoma de Barcelona, Campus de Bellaterra, Spain*

Abstract

JXTA is an open-source initiative that allows to specify a set of collaboration and communication protocols which enable the creation and deployment of peer-to-peer (P2P) applications. This paper provides a survey on its current state regarding the topic of security. The study focuses on the security evaluation of standard peer operations within the JXTA network, highlighting which issues must be seriously taken into account in those applications sensitive to security.

Key words: peer-to-peer, security, peer group, analysis, evaluation, distributed systems, JXTA, Java.

1 Introduction

Peer-to-peer environments provide a virtual network where all involved parties collaborate to supply basic services, such as content sharing, processing or messaging. It is also assumed [35] that all peers have equivalent capabilities, and a central server with more processing power is no longer necessary, ensuring a high degree of decentralization and autonomy to participants. Such environments have become highly popular in recent times due to its great potential to scale and the lack of a central point of failure.

⋆ This work is partially supported by the Spanish Ministry of Science and Innovation and the FEDER funds under the grants TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 ARES.

* Corresponding author.
Email addresses: jarnedo@uoc.edu (Joan Arnedo-Moreno), jherrera@deic.uab.cat (Jordi Herrera-Joancomartí).

Just as the popularity of P2P applications has risen, concerns regarding their security have also increased, specially since it is no longer possible to trust a central server which capitalizes all security operations. As P2P applications move from simple data sharing (such as Gnutella [21] or BitTorrent [13]) to a broader spectrum, they become more and more sensitive to security threats and it becomes capital to take into account which security mechanisms exist in current P2P platforms before deploying them.

JXTA [22] (or "juxtapose") is a set of open protocols that enable the creation and deployment of P2P networks. JXTA protocols enable P2P applications to discover and observe peers, allow communication between them, as well as advertise and locate resources within the network. Such protocols are generic enough so they are not bound to a narrow application scope, but are adaptable to a large set of application types. For that reason, they are implementation independent, so they can be deployed under any programming language or set of transport protocols.

As the popularity of JXTA has increased, not only it has become one of the main testbeds for P2P applications [40] (over 2,700,000 downloads, 120+ active projects, 18,000+ collaborators), but different companies already have taken advantage of the provided framework in order to develop products based on a P2P model [45]. Some of them are quite important ones, such as Verizon or Nokia. The former uses JXTA in a program which directs both voice and data traffic through its network, providing users with an advanced version of instant messaging with telephony tools. The latter uses a JXTA-based software system in its data centers as a means to network monitoring and management. Other companies which use JXTA in their products encompass Codefarm [12], which uses JXTA to distribute its artificial intelligence algorithms for solving problems, such as market financial analysis, across many resources, and SNSing, China's most used social networking software.

This paper provides a survey of the current state of security in JXTA. The survey analyses how JXTA can be affected by typical P2P network threats and it also describes how protection against them can be achieved. The study has been performed analyzing basic peer operations but regarding them not as in an isolated way, but taking into account the whole peer life cycle. Furthermore, the analysis also takes into account one of JXTA's most particular trait: peer groups.

1.1 Related work

A lot of research efforts in the field of P2P have mainly focused towards strictly functionality issues such as scalability [14], efficient message propaga-

tion across the network [49] or access to distributed resources [31]. At present time, the maturity of P2P research field has pushed through new problems such us those related with security. For that reason, security starts to become one of the key issues when evaluating a P2P system and it is important to determine which security mechanisms are available, and how they fit to every specific scenario. Even at the cost of some impact on performance, a security baseline must be kept in any P2P system in order to ensure some degree of correctness even when some system components will not act properly.

Comprehensive reviews regarding security issues in P2P systems can be found in [44,46]. Systems are compared from two different standpoints. On one hand, P2P security is examined based on system capabilities, such as routing or data storage, analyzing the security threats they imply. On the other hand, another approach is to perform a security analysis based on individual security goals, such as reputation, availability or access control. Some attempts to measure the security degree of a P2P system have also been proposed. For instance, in [37] a security framework for such purpose is proposed, but only regarding the hash table based resource location and distribution.

Those generic P2P system security reviews analyze system capabilities or security goals in an isolated manner and for that reason, a more accurate security analysis, for instance based on the peer's life cycle, can not be performed.

By focusing on a single P2P system, instead of providing a generic approach were several ones are reviewed at once, it is possible to present a much more detailed account of every security aspect. In particular, such detailed analysis may assess security solutions as a global layer across all basic peer operations, clearly showing how such operations interrelate and which constraints arise on regards to the deployment of security mechanisms. Unfortunately, such detailed analysis must be performed for each P2P platform independently since the specifics of every P2P system imply different implementations and different security solutions.

Regarding the JXTA P2P middleware, although the platform has been available for several years, studies have been mainly concerned with performance [16,24,3] but not many have discussed such a sensitive issue as the development of secure applications. The most complete effort to present available security mechanisms in JXTA can be found in [50]. However, it can be mainly considered as a statement of design goal achievements. It does not provide a thorough review of all available security mechanisms or a clear idea of how they really work, making it difficult to assess in detail which are the real vulnerabilities in JXTA. On the other hand, JXTA documentation on regards to security is scarce and, in many cases, source code has to be directly reviewed in order to understand how security mechanisms in JXTA really work. Consequently, the results of this study will benefit security-aware platform

developers and designers which want to create JXTA applications, by providing them an up-to-date detailed list of which security related issues should be taken into account. JXTA users may also benefit by realizing which may be the limitations of their applications on the scope of security, so they may take additional measures in order to guarantee a completely secure environment.

Finally, it is worth mention that, to our best knowledge, there is no security evaluation of any other P2P platform regarding peers life circle, similar to the proposed approach with JXTA. For that reason, it is not possible to compare the JXTA security level with other platforms until such detailed analysis has been performed for every platform. Obviously, the detailed security analysis of other P2P platforms and the comparison with the JXTA secure mechanisms analyzed in this paper is completely out of the scope of this review.

1.2 Paper organization

The paper is organized as follows. In section 2, an overview of the JXTA platform is provided in order to understand its main characteristics and methods of operation. Following, in section 3, the basic evaluation model is described by categorizing basic peer operations and threats under the JXTA model. Section 4 presents the security analysis. This analysis includes both core JXTA security mechanisms and additional existing security improvement proposals that are not currently integrated into the base JXTA framework. The final conclusions and directions for new research are outlined in section 5.

2 An overview of JXTA

This section provides a general overview of the main JXTA concepts and components in order to understand the peer operations explained in section 3 and their security concerns. A detailed description of JXTA can be found in [41,42].

The fundamental JXTA architecture is divided into three distinct layers, as shown in Figure 1.

The *Core* layer contains the minimum and essential characteristics, common to all P2P networks. Such operations include peer creation, discovery and communication (even when behind firewalls or NAT), as well as basic security services.

The *Services* layer includes all network services which are not absolutely necessary, but provide desirable capabilities such as resource search, indexing,

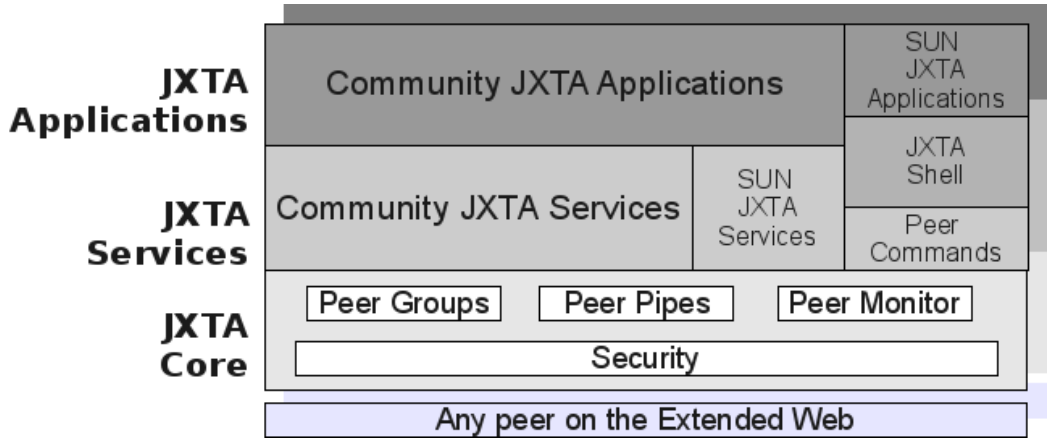

Fig. 1. JXTA's layered architecture

storage and sharing.

The top layer is the *Applications* one, which includes any application deployed using the JXTA framework, such as instant messaging, file sharing or content management.

Notice that the distinction between services and final applications may not always be clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers to choose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers.

2.1 Peers

Each peer in the JXTA virtual network is identified by a unique Peer ID, operating in an independent and asynchronous manner regarding other peers. However, some dependencies may exist depending on which roles they partake.

Usually, peers act as *edge peers*, which could be considered the standard peer type in any desktop application on most devices. They implement the JXTA Core and Services layers as shown in Figure 1 and may interact with any JXTA protocol. Edge peers may also partake two additional roles in order to avoid some specific network constraints: *minimal* and *proxy*. This decision usually depends on its hardware or bandwidth capabilities.

Devices with specific resource constraints (memory, CPU) may act as *minimal peers*, which only implement the JXTA Core layer. They are usually simple devices such as sensors or domotics. In order to use any necessary service to operate within the network, they must rely on *proxy peers*. A proxy peer

summarizes and answers requests on their behalf.

A very important role is that of *rendezvous peers*, which maintain a global index of available resources and help other peers find network services. They also act as beacons which newly connected peers may use in order to join the network. For that reason, rendezvous peers are usually well-known ones, with a DNS name or a static IP address.

Connectivity between peers physically separated from the JXTA network, because of firewalls or NAT, is achieved by means of *relay peers*. They provide the ability to store and resend messages for unreachable peers and, by exchanging route information, message transport across several relay peers is possible in a transparent manner. Nevertheless, peers always attempt direct connections before using a relay.

2.2 Protocols

As explained, JXTA defines a set of protocols (six, specifically) which enable the deployment of P2P networks. Using these protocols, peers may collaborate in a fully autonomous manner by publishing and discovering available resources within the network. Peers may also cooperate in order to route messages, allowing full communication, without the need for them to understand or manage different network topologies.

All JXTA protocols are asynchronous and based on a request/response model, which means that for any given request, zero, one or many responses may be received. A brief description of each protocol is given:

- The *Peer Discovery Protocol (PDP)* allows peers to publish their own resources and make them available to other peers. Any kind of peer may send PDP messages. This protocol is the default discovery protocol, but it is possible for applications to implement and deploy their own protocols.
- In order to obtain information about other peers, the *Peer Information Protocol (PIP)* is used. Using this protocol, it is possible to query peer capabilities or monitor its behavior.
- Peers use the *Peer Resolver Protocol (PRP)* in order to send requests to one or several peers and manage responses. The PRP protocol uses an unique ID associated to every request, which is included in messages. Other core protocols, such as PDP, make use of PRP in order to create its own requests.
- The *Pipe Binding Protocol (PBP)* establishes virtual communication channels between peers named *pipes*, acting as abstract endpoints above any physical network.
- Routes between peers are found with help of the *Endpoint Routing Protocol (ERP)*. Whenever some peer is about to send a message to a destination

but does not know any path, an ERP message is sent to other peers asking whether they know a path.
- Finally, the *Rendezvous Protocol (RVP)* is responsible for the efficient propagation of messages within a group of peers, allowing peers to connect to services and exchange messages.

2.3 Resource publication

Any kind of resource available within the JXTA network, including peers, peer groups, pipes or services, is described by an *advertisement*.

Advertisements are XML documents containing an unique ID and all information regarding that resource and how it may be accessed and exchanged between peers using the JXTA protocols. Peers cannot access a resource without previously retrieving its associated advertisement. Every peer maintains a local cache where all received advertisements are stored for a later use. The local cache directly makes use of the peer's file system in order to organize its content via directories and files.

A sample advertisement is shown in XML Listing 1.

XML Listing 1 - Peer Advertisement

```
<xs:element name="PA" type="jxta:PA"/>

<xs:complexType name="PA">
  <xs:sequence>
    <xs:element name="PID" type="jxta:JXTAID"/>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="Name" type="xs:string" minOccurs="0"/>
    <xs:element name="Desc" type="xs:anyType" minOccurs="0"/>
    <xs:element name="Svc" type="jxta:serviceParams"
                minOccurs="0" maxOccurs="unbounded"/>
  <xs:sequence>
</xs:complexType>
```

Any peer may publish an advertisement to announce the availability of a particular resource by using two different methods: local and remote publication.

In *local publication*, the advertisement is indexed and stored the peer's local cache. Following, the advertisement's index is pushed to a rendezvous peer and is then distributed and replicated between all rendezvous in the global supernetwork peers, using the Shared-Resource Distributed Index (SRDI) service [43,1]. The rendezvous network acts as a remote index cache.

By using this method, it is possible for peers outside the local network (out of broadcast range) to retrieve group advertisements by asking a rendezvous peer. It also enables peers which were off-line for some time to retrieve advertisements published during its disconnection. Whenever a peer receives the

advertisement, the latter is indexed, stored into the local cache and assigned an expiration date.

The advertisement retrieval mechanism is outlined in Figure 2.

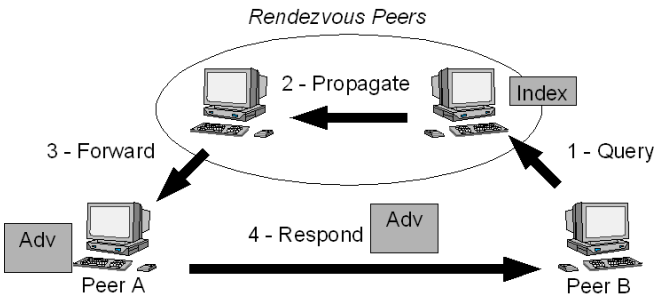

Fig. 2. Advertisement retrieval from Rendezvous peers

It must be remarked that during the publication process, the original advertisement is always kept in the peers' local cache, only its index is distributed. This means that in case the peer goes offline, the advertisement will become unavailable. That makes sense, since also the resource the advertisement publicizes will be unreachable.

In *remote publication* not only indexes are distributed, but the full advertisement itself via the JXTA propagation mechanism. This method is useful in case that the advertisement must be reachable even when the publishing peer is offline. However, under the remote publication method, no assumptions can be made about which peers will really store the advertisement and for how long.

In both methods, when the expiration date is reached, the advertisement is considered stagnant and flushed from the cache, unless the same advertisement is received again, which renews its expiration date. Advertisements may be periodically retransmitted in order to attain permanency or update parameter changes.

As can be seen, advertisement publication and discovery are very important steps in JXTA's peer operation.

2.4 Messaging

JXTA peers use *pipes* in order to exchange messages and access available services. JXTA messages are XML documents with ordered message elements which may contain any type of payload. Messages are the basic data exchange unit in JXTA and all protocols are defined as a set of messages exchanged

between peers.

Pipes provide an asynchronous, unidirectional and unreliable communication channel by default. However, bidirectional reliable channels may be provided on top of them. They offer two operation methods: *unicast* pipes, which allow one-to-one communications, and *propagation* pipes, which allow one-to-many.

JXTA pipes are an abstraction and are not bound to a specific IP address or port. They have a unique ID and are published just like any resource in the JXTA network, so any peer may update them whenever its physical location changes. Both input pipes, used for message reception, and output pipes, using for message sending, are considered *pipe endpoints*, an actual destination in the physical network. Endpoints are dynamically bound to peers via the PBP protocol.

2.5 Peer groups

JXTA 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 [42]. 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 for peers to publish and access 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 [25], 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 only peer group members.

Peer groups are published, discovered and accessed just like any other resource in the network, by means of advertisements.

In order to allow peer group management, JXTA defines the basic primitives for group membership and access control: the Membership and the Access Service. Both are core services which make use of the base JXTA protocols specification in order to achieve their ends. However, JXTA only defines the primitives, while specific applications may implement their own Membership and Access Services depending on their needs.

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 peer group in order to allow other members to know who

is making a request. With this information, the JXTA Access Service may evaluate the credential when a service is accessed and decide whether the request will be granted or denied.

A peer establishes its credential within a peer group by successfully *joining* 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 some 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:

- In the *setup* step, the peer chooses which authentication method will be used for the whole process. If all parameters are correct and the choice is feasible, the peer receives an Authenticator from the Membership Service.
- Following, in the *application* step, the peer completes the Authenticator with all necessary information and tests its correctness. 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.

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

The Access Service provides a single primitive in order to check a credential for a privileged operation. The possible results are disallowed (access denied), permitted (access granted), permitted but expired (the operation would be permitted but the credential has expired) and undetermined (unrecognized credential).

3 Security evaluation model

The first step in order to assess the security degree of JXTA applications is to identify which is the standard peer lifecycle, so that it becomes clear which are the most common operations and, consequently, which deserve better attention on regards to security. Once such operations have been identified, a list of usual security concerns in P2P environments is provided. Our security evaluation model will be based in the cross-reference of standard operations and such

threats, in order to evaluate how the system is protected against each one.

3.1 Standard peer operation cycle

This section describes the standard peer operations for a peer participating in a JXTA network. The order in which they are presented is a logical one for most scenarios. However, it must be taken into account that such order may vary depending on the peer's role.

3.1.1 Platform startup

This is the initial step in order to setup the platform in the physical device which will hold a JXTA peer. This process mainly consists in loading the required libraries and creating the necessary data structures for network connectivity.

At startup, all peers automatically join a default bootstrap peer group named *netPeerGroup*. This peer group is well known to all peers, since it's ID is hard coded into the platform distribution. Peers may decide to stay only in this group or join others once they are connected to the JXTA network. At this stage, edge peers may also try to reach relay or rendezvous peers depending on their local configuration.

3.1.2 Peer group joining

A peer will join those peer groups formed by peers it wants to interact with. This step usually follows startup, so all later operations are only within the boundaries of those specific peer groups and not the whole network. The peer locates the peer group advertisement and joins it via the peer group's Membership Service. In the case that such peer joins the group for the first time, it must locate the peer group advertisement via PDP. From that point onward, the advertisement is already stored in its local cache.

A peer may join several groups, which means that this operation may be performed several times.

3.1.3 Resource publication

Any resource that peer holds and wants to make available to the rest of peer group members is announced by creating and publishing an advertisement as described in section 2.3. This step includes announcing its own presence, by

publishing a peer advertisement, or creating a new group, by publishing a peer group advertisement.

3.1.4 Resource discovery

Available resources in a peer group are discovered by retrieving their advertisements via the PDP protocol. This includes discovering other peers and pipes in order to initiate message exchanges. Usually, pipe advertisements are embedded into other more generic advertisements such as service advertisements.

3.1.5 Message exchange

This would be the most frequent operation during any peer operation cycle. Message exchanges may occur at core protocol level or at service access. At core protocol level, JXTA core protocols are directly used. At service access, two pipe endpoints are established between the communicating peers. An outbound pipe is created by the peer which acts as a client, in order to send requests, and an inbound pipe is created by the peer which acts as a server, in order to receive and process requests.

3.1.6 Disconnection

The peer resigns from all peer groups and goes to offline state in a tidy manner.

This list of operations takes into account some degree of abstraction, since each one actually represents a set of more basic steps. In Table 1, a breakdown of each operation into such basic steps is provided. A more detailed explanation can be found in [42]. Nevertheless, from a security assessment standpoint, the chosen degree of abstraction is enough to provide a clear idea of which are the possible scenarios during any peer's full operation cycle, from startup to disconnection.

3.2 Security threats in P2P networks

The standard security threats in the traditional client/server environment are still valid in P2P environments. Furthermore, the P2P paradigm shift introduces new concerns that must be taken into consideration when designing P2P frameworks. The move from a passive stance (client) to an active one (peer) in the network easily propagates such concern across all its members. Security

Platform startup	Load platform Join *netPeerGroup* Open network listeners Open local cache
Peer group joining	Retrieve group advertisement Instantiate group Fill in Authenticator Join
Resource publication	Create advertisement Local publication Remote publication
Resource discovery	Locate advertisement Store advertisement in local cache
Message exchange	Open pipe Send messages Receive messages Access Service check
Disconnection	Close connections Shutdown platform

Table 1
Basic operation substeps

attacks in P2P systems are classified into two broad categories: passive and active [23].

Passive attacks are those in which the attacker just monitors activity and maintains an inert state. The most significant passive attacks are:

- *Eavesdropping (Evs)*, which involves capturing and storing all traffic between some set of peers searching for some sensitive information (such as personal data or passwords).
- *Traffic analysis (TAn)*, where the attacker not only captures data but tries to obtain more information by analyzing its behavior and looking for patterns, even when its content remains unknown.

In active attacks, communications are disrupted by the deletion, modification or insertion of data. The most common attacks of this kind are:

- *Spoofing (Spf)*, in which one peer impersonates another, or some outside attacker transforms communications data in order to simulate such an outcome.
- *Man-in-the-middle (MitM)*, where the attacker intercepts communications between two parties, relaying messages in such a manner that both of them still believe they are directly communicating. This category includes data alteration between endpoints.
- *Playback or replay (Pb)*, in which some data exchange between two legitimate peers is intercepted by the attacker in order to reuse the exact data at a later time and make it look like a real exchange. Even if message content is encrypted, such attacks can succeed so long as duplicate communications are allowed and the attacker can deduce the effect of such a repeat.
- *Local data alteration (LDA)*, which goes beyond the assumption that attacks may only come from the network and supposes that the attacker has local access to the peer, where he can try to modify the local data in order to subvert it in some malicious way.

Apart from security threats that take into account a malicious attacker, it is also very important to take into account *Software Security Flaws (SSF)* in a security survey. Specifically, which steps are taken by the developers in order to minimize the probability that a bug that may later jeopardize a deployed system.

4 Security evaluation

From the standpoint of basic security requirements which are desirable in JXTA, they are very similar to those of any computer system: confidentiality, integrity and availability. In order to achieve them, these requirements should translate into an architecture that includes authentication, access control, audit, encryption, secure communication and non-repudiation.

JXTA remains neutral to cryptographic providers or security schemes. In its initial conception it does not mandate any specific security solution, providing a generic framework where different approaches can be plugged in. Enough hooks and place holders are provided in order for each specific application to implement its own security solution. Nevertheless, in a present day P2P framework, relying in the fact that each application will build from scratch its own security solution is not enough, since it will usually mean that security will be overlooked, as its is often the case. It is very important that basic tools and functionalities are already available, providing a default degree of security but allowing further modularization if necessary. As such, basic security services (encryption, integrity and authenticity) should be provided, at least, at the Core layer, even though some applications may chose not to use them.

In this section we will analyze whether the current iteration of JXTA (version 2.5) is up to this desiderata for each of peer basic operation and according to the standard threats to P2P networks.

4.1 Platform startup

During the framework startup phase, since the networking capabilities are not operative yet, no threat related to a networked environment applies. However, it does make sense to take into consideration local data alteration on such libraries, since it is at this precise moment that JXTA libraries are loaded into the system.

Due to the open nature of JXTA, the official project page protects software integrity with SHA1 digests [34], in order to avoid malicious distributions. For that reason, a local attack is necessary to actually deploy fake libraries into the system. The current Java distribution also uses code signing to provide a basis for integrity and authenticity checking of installed libraries. Specifically, it makes use of the Java jarsigner [38] tool when the source code is built. The necessary keystore information to sign the code, both the private key and certificate, is distributed with the source code. Unfortunately, this approach allows anybody to sign malicious the code, since it uses a self-signed certificate, and the keystore password is easily available in the build files (it is *jxta.platform*), meaning that the whole keystore content may be easily compromised. This is unavoidable if it must be possible for anybody to build the libraries. The only solution would be to have an official distribution, built by a trusted party, whose keystore information is not widely available.

Apart from data alteration, it is worth analyzing JXTA's libraries security from a Software Security Flaw standpoint. JXTA is an Open Source Software (OSS) project, which is a good indicator when specifically analyzing security [10]. Obviously, security through obscurity should not be applied, so any software security mechanism which depends upon secrecy tends to eventually fail, as bugs or security flaws are discovered. Since JXTA code is public [40], it has been audited by a large number of individuals all across the Internet. The use of an OSS approach not only ensures current security, but allows direct improvement from the JXTA developer community, maximizing the networking effect.

Nevertheless, it is worth mention that opening the source code creates the opportunity for individuals to review security, but it cannot guarantee that such reviews will occur. There is also the fact that no guarantee can be made that a review will find any particular security flaw in a system, but that problem is also common to closed source projects. In any case, OSS allows developers

with security concerns to directly assess whether the JXTA framework is up to their needs.

4.2 Peer group joining

The first step in order to join any peer group is to retrieve its peer group advertisement. The implications of this specific substep will be explained in the next subsection. Therefore, we will focus here on the actual group instantiation and join operation.

As described in subsection 2.5, the Membership Service is the key security mechanism for the group join operation. Through this service, peers claim identities by proving they are the legitimate owner. This service is defined as generic in the JXTA specification, so each application may implement it according to its own needs. However, the JXTA reference implementation, as far as version 2.5 [41], provides three available Membership Services which are ready to use.

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. This service is widely used in applications with no security concerns.

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 freely roams across the overlay network, which makes this method completely insecure. In fact, this group membership service was created as a sample and a means of testing and 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 is the only one that is considered secure and the one that will be analyzed.

PSE provides credentials based on PKIX [11] certificates. As shown in XML Listing 2, any number of such certificates may be included as *Certificate* elements in the PSE credential, together with the Peer Group ID and the subject's Peer ID. The credential itself is also signed.

The authentication procedure in order to join a PSE peer group can be summarized as follows:

XML Listing 2 - PSE Credential XML schema

```
<xs:complexType name="jxta:PSECred">
  <xs:sequence>
    <xs:element name="PeerGroupID" type="jxta:JXTAID"/>
    <xs:element name="PeerID" type="jxta:JXTAID"/>
    <xs:element name="Certificate" type="base64binary"
             minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="Signature" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

(1) The user introduces its personal password.
(2) The peer initializes an Authenticator for the peer group the user wants to join, using the provided password and the peer's ID.
(3) Using this information, an encrypted keystore in the local cache is located and opened, or created if not already existing.
(4) The user's certificate chain is retrieved.
(5) Such certificate chain becomes the group credential for that peer.
(6) The peer may interact with other ones in the same peer group.
(7) The private key in the keystore is used by the peer in secure protocols when needed.

All the enumerated steps in the join process using the PSE Membership Service are completed via local calls to JXTA libraries. For that reason, peer group joining is not concerned with network-based attacks (eavesdropping, traffic analysis, MitM or replay), since there are no real network based operations. It also means that any vulnerability in the join process must be exploited by a local attacker.

As we can see, three actors interact in this process: the final user (the human being in front of the computer screen, or some agent), the peer (the application) and the peer group (JXTA libraries which control group access). That means that two spoofing methods must be taken into consideration:

- Impersonating the user: Unauthorized access to keystore content. This is equivalent to taking control of another user's peer.
- Impersonating the peer: Unauthorized identity claim and credential generation. This is equivalent to being able to claim any identity within a peer group.

In the first case, all security relies on the strength of keystore encryption and its password. Unfortunately, the keystore is stored as a simple file, which may be easily copied and distributed, and no mechanisms exist in order to plug in more advanced methods of key management (such as cryptographic hardware tokens).

Regarding peer impersonation, it is must be pointed out that peers under a

PSE Membership Service are authenticated only by being able to access a local keystore. During the process, the Membership Service is not concerned with the validity of certificate chains (whether it is signed by proper Certification Authorities or not expired), and the certificate content is never checked. Credentials are actually checked during the message exchange operations, as will be explained in section 4.4.

As a result, anybody with access to a private key and a certificate (a self-signed one would be enough) will be able to correctly join any group using PSE and initially claim any identity. For that reason, the default join scenario is easily threatened by peer spoofing attacks, since anybody may create a valid keystore using public domain tools [39]. There is no real security on peer identity claims on regards to the Membership Service. These kind of peers which hold the necessary information in order to generate a correct PSE Peer Group Authenticator, but are not really group members because their certificate is not properly signed are named *interlopers*. In this scenario, they can become an annoyance, but can be spotted and dealt with when accessing services.

A further concern that developers should take into account when using the PSE Membership Service is the fact that no revocation mechanism is provided. JXTA takes into account the possibility that a group member voluntarily resigns from a peer group, but does not provide the capability to expel a group member for some reason such as malicious behavior. No primitives exist within the framework which may somehow allow this. Developers must create their own revocation schemes from scratch.

To sum up, we can see that PSE is a kind of toolbox that allows the implementation of different models based on securing identities via digital certificates, but it does not provide a clear structure of how trust is managed in a peer group (whose signatures are trusted or which peers are allowed to sign certificates). To properly configure PSE, the application must define the real trust anchors and some method that guarantees key authenticity. Under this scenario, trust anchors may take the form of PGP trust chains [20], where every peer may issue certificates, or a Certification Authority (CA), a single entity able to issue digital certificates within the group. In the latter case, it may based on a fully centralized approach [11], where a single peer holds the CA private key, or a distributed one [15], where the private key is split between several peers, which must collaborate to issue certificates.

In scenarios where each peer may generate certificates, PSE may not properly, since it is not possible to easily guarantee key authenticity with the provided primitives. This is a weakness developers of JXTA agree that should be addressed [51]. Even under this assumption, it must be remarked that correctly setting a trust anchor in a pure P2P environment is not an easy task. First of

all, in-band certificate generation procedures are easy prey to MitM attacks. In addition, when peer equality must be preserved, the proposed solution should avoid that the peer group may become entirely reliant on some peer, which gets some extra power above the rest.

One of the original creators of JXTA, Yeager, provides a specific trust model for the membership service in [51], which tries to solve the problems previously described. Without actually recognizing a specific CA for each peer group, he proposes that rendezvous peers become the system's trust anchors, the main reason being that they are well protected. Each edge peer uses rendezvous certificates as root certificates in order to ensure key authenticity. Furthermore, to acquire a certificate the peer must be authorized via an LDAP (Lightweight Directory Access Protocol) [48] directory with a recognized protected password. Rendezvous peers may use a secure connection to the LDAP service to authorize requesting peers. The rendezvous peer's root certificate is securely distributed out-of-band by being directly included into the JXTA libraries.

However, developers should take into account that this proposal is ultimately based in a centralized approach and peer groups become heavily reliant on external entities, which makes the system unfeasible in ad hoc environments. Furthermore, the proposal does not specify who configures or manages the LDAP service. It is assumed that some group administrator will do it, which makes it a logical approach in an enterprise environment, but moves away from a pure P2P model.

4.2.1 Non-core JXTA group membership

Due to its open project approach, the JXTA platform offers additional security mechanisms, not currently included in the standard distribution, that have been proposed in order to tackle some of the described issues, by providing additional Membership Service implementations.

An initial proposal can be found in [30], but its similarity with the Passwd Membership Service inherits most of its pros and cons. For that reason, it cannot be considered completely secure either.

Another proposal [29] is similar to the trust model proposed by Yeager, adding extra capabilities upon the basic Membership Service. This approach is based on a centralized PKI and a basic challenge-response protocol [36] as a means for authentication during the join process. Its main contribution is to provide a method which peers may use in order to authenticate the group itself. It also allows to check peer group membership during the join process and defines a trust anchor based on both an external centralized LDAP server and a sign server. Using this method, it is no longer possible to easily spoof peer identities. The LDAP server provides peer certificate management and the sign server

deals with group authentication, keeping a secure copy of a private key which represents the whole group. Again, developers should take into account that this proposal becomes reliant on external entities and cannot be considered a pure P2P approach.

More elaborated proposals are presented in [52,2], based on joint authorization by multiple peers under voting schemes in order to maximize decentralization. Under this approach, JXTA credentials are signed certificates issued by a group CA, however group access is based on an agreement reached between several group members, instead of being entirely up to the CA's decision. The main difference between both proposals is that [2] includes a rank system, where peers who join the group ("newbies") have the least privileges, but they may rise to higher positions as they contribute to the group.

Finally, a proposal which moves away from a centralized PKI and uses a web-of-trust approach is described in [4]. This model is specifically concerned in providing proof of peer group membership, not peer identity, via trust relationships: whenever peer A creates a trust relationship with peer B, it is vouching for B's group membership to other group members.

The model defines two different sets of peers: 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. Group management is achieved via additional services that take into account the idiosyncrasies of JXTA's messaging capabilities: the Patron Discovery, Sign, Trust Path Discovery and Credential Retrieval services. An advantage of this proposal is that it is not constrained to the JXTA file-based keystore type, using an interface named CryptoManager which arbitrates the join process and acts as a proxy for any cryptographic provider (software or hardware based). This approach also takes into account the fact that not all cryptographic providers are accessed via plain text passwords (for example, using biometrics), providing developers with the capability to choose a suitable degree of security according to each environment.

4.3 Resource discovery and publication

Resources provided by peers in the JXTA network are represented by XML documents named advertisements, as detailed in subsection 2.3. In order to access such resources, their advertisement must be somehow retrieved. Advertisement discovery and retrieval is achieved via message exchange using the PDP and PRP core protocols (see subsection 2.2). Since it is a network-borne

operation, we can focus on all threat types. We will discuss both resource discovery and publication in this section, since both share the same security mechanisms (only data flow direction changes between both operations)

In the current JXTA reference implementation, advertisements may be secured at two different levels: at transport layer and at advertisement layer. Since securing at transport layer means considering an advertisement as a standard message, the provided security methods will be explained in subsection 4.4, where security in messaging is analyzed. In this subsection we will focus on advertisement-specific methods.

At advertisement layer, security is achieved by digitally signing advertisements at application level by using a special type of advertisement named *Signed Advertisement*. The direct use of digital signature on the advertisement makes it possible to support both local and remote publication (via propagation to multiple peers). As a precondition to use this special type of advertisement, it is mandatory to join a peer group that uses the PSE Membership Service, since the necessary cryptographic keys to generate and validate the signature are obtained from the keystore and credential associated to this service. Signed advertisements will only be sent to members of that group. In any peer group which does not use this service, signed messages cannot be exchanged between its members. As a result, any security concern related to PSE is inherited by signed advertisements, such as the lack of real authenticity without setting a trust anchor at application level.

The XML schema definition for a Signed Advertisement is shown in XML Listing 3. It contains the signer's credential (the `PSECred` element, a credential for a PSE Membership Service), the signature and the original advertisement. The `Advertisement` element encapsulates the original XML advertisement as plain text encoded via the Base64 algorithm [26]. The content of the `Signature` element is generated by applying the RSAwithSHA1 algorithm to the original advertisement, XML formatted (not its Base64 encoded form). In order to feed the algorithm, the XML data is processed as plain text. The result is henceforth Base64 encoded in order to be represented as plain text into the XML document. Once a signed advertisement is received by a peer, the signature is validated, the actual advertisement extracted and then stored into the local cache.

XML Listing 3 - Signed Advertisement XML schema

```
<xs:element name="SA" type="jxta:SA"/>
  <xs:complexType name="jxta:SA">
  <xs:sequence>
    <xs:element name="PSECred" type="jxta:PSECred"/>
    <xs:element name="jxta:Signature" type="base64binary"/>
    <xs:element name="jxta:Advertisement" type="base64binary"/>
  </xs:sequence>
</xs:complexType>
```

On regards to advertisements, JXTA does not currently seem concerned with passive attacks, since it offers no advertisement protection against them. They are freely exchanged between peers in plain text. In fact, since they are structured using XML, it is very easy for an eavesdropper to read search for specific content (no need to process binary structured data). A human being can directly interpret advertisements with a text editor.

No effort is made either in order to masquerade advertisement traffic, so it is feasible that an attacker may obtain some interesting information by just analyzing traffic, specifically detecting which peers offer more resources (since they are the ones which publish more advertisements). Using this method, it is possible to find the most interesting peers when looking for potential victims to attack. In fact, since anybody may instantiate any peer group, acting as an interloper, and then discover advertisements bound to the group, this kind of attacks are easy to perform. It is not even necessary to tap the network. How easily advertisements are exchanged is both a bonus for open services and a bane for tight security environments.

If we assume that applications which use PSE correctly set a trust anchor in order to guarantee certificate authenticity, then active attacks such as spoofing, MitM and replay attacks may be correctly countered by digitally signing advertisements. Using this method, false advertisements may still occur within the peer group, but because of the non-repudiation property of digital signatures, it is easy to pinpoint offenders.

As far as resource publication is concerned, since every peer is completely reliant on its rendezvous peer in order to properly distribute the advertisement index to the rest of the network, and no control is made about which peer may become a rendezvous one, it is possible to pull off MitM attacks by masquerading as a one. No control mechanisms exist either to automatically detect a misbehaving rendezvous peer. For that reason, each application should always deploy some method in order for peers to be able to identify real rendezvous peers.

Finally, since secure advertisements lose the signature when stored into the local cache, the threat of a local attacker still exists, since it is possible to modify the local cache content, inserting or modifying false advertisements which redirect group members to false services in malicious peers.

4.3.1 Non-core advertisement security

Another approach to advertisement security which is not included in the core JXTA distribution, exists in [5]. The proposed method is entirely based on XML Signature [47] (xmldsig) and provides advertisement authenticity and non-repudiation in a lightweight manner, without the need of a trust anchor.

This is achieved by using Crypto-Based Identifiers [33] (CBIDs), which will be detailed in the following section.

The main contributions of this proposal are:

- Keeping interoperability by maintaining the advertisements base format, instead of creating a completely new one. This approach makes it possible to seamlessly integrate peers which support advertisement security which those who do not (or simply chose not to for some reason).
- Providing a method to efficiently publish CBID-key binding by taking advantage of JXTAs own capabilities, without the need of additional protocols.
- Maintaining end-to-end advertisement security for its whole lifetime, not just during transport, providing protection against local data alteration attacks, in contrast with the core signed advertisement approach

A sample of the XML signed advertisement used in this proposal is shown in XML Listing 4 (some ID's and Base64 encoded data have been shortened to improve readability).

XML Listing 4 - Xmldsig based signed Advertisement

```
<jxta:PGA xmlns:jxta="http://jxta.org" xml:space="preserve">
  <GID>urn:jxta:uuid-2AD...5F02</GID>
  <MSID>urn:jxta:uuid-EB76C91C2A0F49F685C...B3812F206</MSID>
  <Name>SampleGroup</Name>
  <Desc>Signed Peer Group Advertisement</Desc>
  <ds:Signature
      xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm=
          "http://www.w3.org/TR/2001/REC-xml-c14n-20010315">
      </ds:CanonicalizationMethod>
      <ds:SignatureMethod Algorithm=
          "http://www.w3.org/2000/09/xmldsig#rsa-sha1">
      </ds:SignatureMethod>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/
              2000/09/xmldsig#enveloped-signature">
          </ds:Transform>
          <ds:Transform Algorithm=
              "http://www.w3.org/2001/10/xml-exc-c14n#">
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod Algorithm=
        "http://www.w3.org/2000/09/xmldsig#sha1">
        </ds:DigestMethod>
        <ds:DigestValue>Ko0R31wMpcJ17VAmtaUf7nS/KU4=
        </ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> nloCRUg4WB0H+DcEAuLKGYhqvsfdRCy4R...
      ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
    </ds:SignatureValue>
    <ds:KeyInfo>
      <KeyName>urn:jxta:uuid-596162...BCF0646C103</KeyName>
    </ds:KeyInfo>
  </ds:Signature>
</jxta:PGA>
```

In the current JXTA reference implementation, messaging has been secured under the assumption that the Personal Security Environment (PSE) has been chosen as a group's Membership Service. By using the PSE, JXTA messages may be secured at two different levels in core messaging: at the messaging level, by using the CBJX [7] protocol, and at the wire transport level, via its own definition of TLS [17]. The messaging level operates at a higher abstraction level, encapsulating data without knowledge of the real network topology. The encapsulated data is then sent via the transport level, which does take into account both network topology and available transport protocols at the peer's node.

The standard messaging level provides the capability to include any type of digital signature elements into messages to be sent across the network. However, current standard messaging protocols never make use of this feature. CBJX (Crypto-Based JXTA Transfer) is a JXTA-specific protocol which provides lightweight secure message source verification by including into messages its own self-defined digital signature element, providing data integrity and authentication. This approach provides protection against active threats.

Even though CBJX is formally specified as a wire transport protocol, it can be truly considered to operate at the messaging level, or, more exactly, at a meta-messaging level. The main reason is that it lacks the capability to directly send messages between endpoints, which is what ultimately defines a wire transport protocol in JXTA. CBJX pre-processes messages in order to provide an additional secure encapsulation, creating a new message that is then relayed to an underlying wire transport protocol. For that reason, we classify CBJX as message level security.

In addition to the original message's digital signature, an information block, according to the definition shown in XML Listing 5, is also encapsulated with the secured message: a `CbJxMessageInfo` element, which contains the source peer credential (a PSE certificate), both the source and destination addresses, and the source peer ID.

This cryptographic information block is digitally signed as well, generating two distinct signatures within the final CBJX message. The certificate inside the cryptographic information block is used to validate both signatures.

In order to generate both signatures, XML data is serialized and then fed to the signature algorithm, processed as plain text. An overview of the final message encapsulation is shown in Figure 3. CBJX encapsulates signatures by using a single `Signature` element containing a Base64-encoded PKCS#7 [27] binary signature. Once the final CBJX message is complete, it is sent using

XML Listing 5 - CBJX crypto-information XML schema

```
<xs:complexType name="cbjx:CbJxMessageInfo">
  <xs:sequence>
    <xs:element name="PeerCert" type="xs:base64binary"/>
    <xs:element name="DestinationAddress" type="xs:string"/>
    <xs:element name="SourceAddress" type="xs:string"/>
    <xs:element name="SourceID" type="jxta:JXTAID"/>
  </xs:sequence>
</xs:complexType>

<xs:simpleType name="JXTAID">
  <xs:restriction base="xs:anyURI">
    <xs:pattern value="([uU][rR][nN]:
                        [jJ][xX][tT][aA]:).+\-.+"/>
  </xs:restriction>
</xs:simpleType>
```

any wire transport protocol, just like as a standard message.

On reception, the CBJX information block is extracted and both signatures are validated, acting in a transparent manner as far as upper layer protocols is concerned by providing the original message.

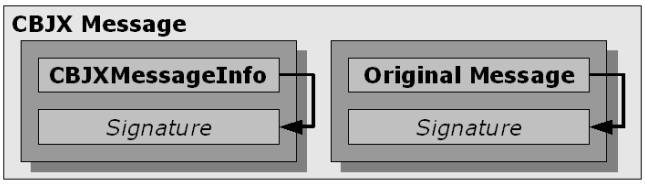

Fig. 3. CBJX secure encapsulation

Apart from digital signatures, CBJX provides an additional lightweight authenticity method by using CBIDs [33]. The concept of CBIDs, or statistically unique and cryptographically verifiable IDs, was initially conceived for IPv6 addressing in order to solve the issue of address ownership, avoiding router supplantation attacks and binding update packet spoofing [6,8]. Using this mechanism, each address is automatically bound to a specific node. It is important to notice that by using this method, authentic messaging is provided without the need of certificates issued by a trust anchor.

At wire transport level, JXTA provides its own definition of standard TLS, allowing private, mutually authenticated, reliable streaming communications. Thus, TLS protects against both passive and active threats. As a wire transport protocol, it is responsible for encoding message data and sending it across the network.

The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol provides connection security with two basic properties:

- The connection is private. Symmetric cryptography is used for data encryp-

tion (e.g., DES [18], RC4 [28], etc.) The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by the TLS Handshake Protocol. The Record Protocol can also be used without encryption.
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

In the specific case of JXTA, messages are delivered securely between endpoints even when multiple hops across peers are necessary.

Even though TLS is a binary protocol, JXTA implements some of its data exchanges using XML elements (which encompass binary content). Three element types are defined in order to implement the protocol: TLS Content, which encapsulates transmitted secure data, Acknowledgements, which acknowledge data reception, and Retries, when a message is sent because of an apparent failure at a previous transmission. The latter element will be always present with a TLS Content element. All standard binary data structures defined in TLS are included into the TLS Content element.

By combining both CBJX and TLS, it is possible to trump both passive and active attackers by achieving data privacy, integrity and authenticity. Application developers may decide which protocol to use depending on their constraints (such as operating under a non-PSE peer group). It also must be taken into account that both types of transport methods do not support full advertisement propagation, they only support point-to-point communications. That means that applications which are based on multicast are still prone to security threats.

4.4.1 Service access

When accessing a service peer credentials must be checked in order to decide whether some peer has real access to that service. This is necessary since, as we could see in subsection 4.2, actually everybody may instantiate a peer group and try to access resources, acting as an interloper. This may be achieved in JXTA by using the peer group Access Service.

As far as the Access Service is concerned, the current JXTA reference implementation offers three kinds of access control, each one bound to each different membership service credential type:

The *Always* Access Service, which does not really check for access control and allows 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 each group operation. 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 operation is permitted or not.

It must be pointed out that current Access Service approaches are strictly tied to ensure that some identity may access some service. Whether that identity really belongs to a legitimate peer group member is never checked, it is always assumed correct. Since the Membership Service is not up to the task of checking group membership either (any peer may claim any identity), as exposed in section 4.2, this is something JXTA application developers should heavily take into account. However, it is possible to implement an Access Service which also checks group membership, as it is the case of the non-core approach in [4], previously presented in subsection 4.2.1.

Furthermore, the Access Service provides a single primitive which just checks credential content, but does use on any kind of authentication protocol. This is not sufficient to guarantee protection against spoofing, since credentials are freely exchanged across the network, being public. Some other method must exist which tests credential authenticity, such as TLS or CBJX, in order to guarantee authenticity.

4.5 Disconnection

No real vulnerabilities threaten disconnection, apart from those which force an unintended shutdown due to unauthorized local access to the application. However, such problems are related to the operating system, so it can be considered outside the scope of this study. Disconnection was mainly included for the sake of completeness in formalizing the peer's full lifecycle.

4.6 Security evaluation summary

Table 2 summarizes the JXTA security evaluation, as far as core functionalities is concerned (non-core proposals are not included). For each JXTA basic operation, it is shown whether it is vulnerable to each of the typical threats and which security mechanism exists in order to counter it. The threats are those listed in section 3.2: Eavesdropping (Evs), Traffic Analysis (TAn), Spoofing

(Spf), Main-in-the-Middle (MitM), Replay attacks (Rp), Local data alteration (LDA) and Software security flaws (SSF).

Operation\Threat	Evs	TAn	Spf	MitM	Rp	LDA	SFF
Startup	N/A	N/A	N/A	N/A	N/A	V(1)	P(OSS)
Join	N/A	N/A	V(5)	N/A	N/A	P(Key Enc.)	P(OSS)
Publish/Discover	V(2)	V(3)	P(Signed Adv.)			V(4)	P(OSS)
Messaging	P(TLS*)	V(3)	P(TLS*/CBJX)			V(4)	P(OSS)
Disconnect	N/A	N/A	N/A	N/A	N/A	N/A	P(OSS)

Table 2
Security in JXTA summary (N/A: Non-applicable. V(type): Vulnerability exists. P:Protected(mechanism). *Not usable for message propagation)

According to Table 2, the main vulnerabilities in JXTA's basic peer operations may be summarized in five different types as follows:

V(1): Code signing may be easily compromised. Malicious executable code can easily be built and cannot be automatically discovered when installed.
V(2): No encryption mechanism exists. Advertisements are transmitted in plain text.
V(3): No data flow masquerading mechanism exists. It is easy to identify important peers by its traffic.
V(4): No integrity check is enforced on the local cache. Changes by a local attacker are not discovered.
V(5): No real authentication is enforced. Any peer may ultimately join any group as an interloper.

4.7 Security cost analysis

This subsection provides a brief overview of the overhead introduced by the security mechanisms detailed in this survey. In order to asses the impact on performance, two different sets of scenarios have been taken into account when running a set of experiments: advertisement publication and discovery, evaluating the Signed Advertisement mechanism, and message propagation, evaluating CBJX. We have focused exclusively on CBJX since thorough studies on JXTA's TLS performance already exist [16].

A set of 50.000 tests have been run for each different scenario using a PC with a 1.20 GHz Intel Pentium M processor and 1 Gb of RAM under Ubuntu 8.10 and SUN's Java Runtime Environment version 1.6.0.10 (which includes the Java Cryptographic Extension (JCE)) . We decided to use a computer

which is below today's average standards to assess the impact of using JXTA's security mechanisms on lower end machines.

In the tests, messages of different sizes have been processed in order to measure the time needed for secure encapsulation (in the cases of advertisement publication and message transmission) and the time for extraction (in advertisement discovery and message reception). Different suggestions exist on regards to desirable data size in order to test message performance [16,3]. We have decided to chose the one in [16]: 1kb, 10Kb, 100Kb, 1Mb and 10Mb. No suggestions exist for advertisement size, but by analyzing their XML schema [42], we consider that a good estimation is about 1Kb for small sized ones, 3Kb for medium sized ones and 5Kb for the bigger ones.

The total time for processing protected data (encrypted and/or signed) has been divided in three separate categories, namely crypto processing, XML processing and data processing. The first category measures the time needed to compute cryptographic operations, for instance digital signatures, validation, encryption, etc. The second category, XML processing, measures the time spent to process XML structures in order to provide secure encapsulation/extraction. Finally, data processing measures the time needed for raw data operations, mainly data serialization and Base64 encoding/decoding.

The time spent in cryptographic operations mainly depends on the efficiency of the cryptographic provider or even the use of hardware implementations. As we have already mentioned, in our tests we have used the Java Cryptographic Extension (JCE) as a default cryptographic provider. The time dedicated to XML processing depends on the underlying XML parser (in this case, JXTA uses its own XML processing library). Finally, data processing is usually dependent on raw computer power (memory and input/output speed).

The experiment results are presented in Table 3 regarding securing advertisements and Tables 4 and 5 regarding securing messages. Data shows that cryptographic operations are the heaviest burden when using JXTA's security mechanisms. Furthermore, it is worth mention that the sender needs more processing time than the receiver to deal with cryptographic data.

In order to provide a better idea of the magnitude of the time for securing data, measurements have been taken for processing time when no security is used. The estimated time for generating advertisements and messages is about $4ms$ and the time needed for reading discovered advertisements is about $2ms$. These values are very small compared to the cost of cryptographic operations, such as $455ms$ in the worst case scenario (securing a 10Mb message). However, regarding a real scenario, we should have into account the required trip time between the communicating peers. This variable is dependant on many factors, but from the measurements in [16], its value goes from $100ms$ for 1Kb up to

Operation	Publication			Discovery		
Advertisement Size	1Kb	3Kb	5Kb	1Kb	3Kb	5Kb
Crypto processing	36.295	37.953	39.335	2.199	2.637	3.04
XML processing	0.318	0.518	0.699	0.621	1.266	1.877
Data processing	0.422	0.943	0.74	0.032	0.032	0.032
Total	37.035	39.414	40.774	2.852	3.395	4.949

Table 3
Average processing time for securing advertisements. Time in *ms*, up to three decimals

Operation	Transmission				
Message Size	1Kb	10Kb	100Kb	1Mb	10Mb
Crypto processing	34.277	44.758	51.33	99.96	455.579
XML processing	0.169	0.175	0.186	0.201	0.207
Data processing	0.061	0.062	0.076	0.1	0.11
Total	34.507	44.995	51.592	100.261	455.896

Table 4
Average processing time for securing messages. Time in *ms*, up to three decimals

Operation	Reception				
Message Size	1Kb	10Kb	100Kb	1Mb	10Mb
Crypto processing	0.969	1.315	4.764	41.051	394.623
XML processing	0.275	0.277	0.295	0.296	0.395
Data processing	0.009	0.013	0.066	0.573	6.001
Total	1.253	1.605	5.125	41.92	401.019

Table 5
Average processing time for validating secured messages. Time in *ms*, up to three decimals

$10.000ms$ for 10 Mb messages. Taking this values, the magnitude of the time for secure data processing is affordable regarding the total time, which includes processing and trip time.

5 Conclusions and further work

Being an OSS project, JXTA has been intensely reviewed, and as a result, its security features have improved over time. It can be summarized that the current implementation of JXTA has evolved to include an acceptable level of security, fulfilling minimum requirements for present day applications. However, this is at the cost of being bound to a very specific group membership model: PSE. In the case that a custom model is chosen for some applications, we will find out that most of its security capabilities may no be directly used, only CBJX becomes still available. This is not always desirable in a framework that was conceptualized to be open and easy to adapt to any environment. It would be useful that any custom application security model could make use of as many as possible JXTA secure mechanisms such as TLS or advertisement signature. Another useful feature, right now constrained by the assumption that PSE will always be used, would be the capability to use different types of keystores, apart from that in each peer's the local cache. Specially, being able to go beyond using the file system as cryptographic storage. However, it is not possible for now.

It is also important to take into account when designing JXTA applications that, even though PSE provides a certificate-based secure environment, it is still necessary to chose some methods in order to guarantee key authenticity. PSE assumes no trust model, just provides the necessary tools in order to deploy it. Furthermore, it makes no effort to provide any method of group membership revocation.

Finally, core JXTA still has some gaps pending to be filled even when all its security capabilities are fully used (see Table 2). First of all, no mechanism exists in the current version of JXTA in order to secure messaging for propagation mechanisms, specially one that provides some degree of privacy. In addition, no security exists for the local cache, even though very important data is stored inside. At least, some degree of integrity would be desirable, such as maintaining advertisement signatures. Fortunately, due to its open project approach, additional security proposals, though not currently included in the standard JXTA distribution, have been proposed in order to solve all of these issues.

Once the state of security for JXTA has been clearly established, further research includes providing additional security services at core level without the need of PSE. Our main efforts will be twofold. First of all, creating a peer group environment where membership is really checked during the join process. Then, improving advertisement and messaging security, both in cache storage as well as during transport. Additional research may include comparing the current state of security of JXTA to that of other P2P middlewares, from both

a qualitative and quantitative standpoint.

References

[1] Abdelaziz, M., Traversat, B. et al, 2003. Project JXTA: A loosely-consistent dht rendezvous walker.

[2] Amoretti, M., Bisi, M., Zanichelli, F., Conte, G., 2005, Introducing secure peer groups in SP2A, Proceedings of the 2nd IEEE International Workshop on Hot Topics in Peer-to-Peer Systems, co-located with Mobiquitous 2005m pp. 62–69.

[3] Antoniu, G., Hatcher, P., Jan, M., Noblet, D. A., 2005. Performance evaluation of JXTA communication layers, CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 1, 215-258

[4] Arnedo-Moreno, J., Herrera-Joancomartí, J., 2006. Providing collaborative mechanism for peer group access control, in: Proceedings of the Workshop on Trusted Collaboration, IEEEPress, pp. 1–6.

[5] Arnedo-Moreno, J., Herrera-Joancomartí, J., 2008. Persistent interoperable security for JXTA, in: Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC) 2008, IEEEPress, pp. 354–359.

[6] Aura, T., 2005. Cryptographically Generated Adresses (CGA), `http://www.ietf.org/rfc/rfc3972.txt`.

[7] Bailly, D., 2002. CBJX: Crypto-based JXTA (an internship report) 108–109.

[8] Bononi, L., Tacconi, C., 2007. Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks, Int. J. Inf. Secur. 6 (6) 379–392.

[9] SUN Microsystems, Java Cryptography Extension (JCE), `http://java.sun.com/javase/technologies/security`

[10] Caloyannides, M., Landwehr, C. et al, 2001. Does open source improve system security?, Software IEEE 18 (5) 57–61.

[11] CCITT, 1998. The directory authentication framework. Recommendation, `http://www.nist.fss.ru/hr/doc/mstd/itu/x509.htm`.

[12] Codefarm Inc., Codefarm, `http://www.codefarm.com`.

[13] Cohen, B., 2003. Incentives build robustness in bitTorrent, 1st Workshop on the Economics of Peer-2-Peer Systems.

[14] Dai, L., Cao, Y., Cui, Y., Xue, Y., 2009. On scalability of proximity-aware peer-to-peer streaming, Comput. Commun., volume 32, 144–153

[15] Desmedt, Y.G., Frankel, Y., 1989. Threshold cryptosystems. CRYPTO '89: Proceedings on Advances in cryptology 307–315.

[16] Deters, R., Halepovic, E., 2005. The JXTA performance model and evaluation, Future Generation of Computer Systems 21 (3) 8377–390.

[17] Dierks, T., Allen, C., 1999. The TLS protocol version 1.0, `http://www.ietf.org/rfc/rfc2246.txt`.

[18] FIPS - Federal Information Processing Standard, 1977. Data Encryption Standard, National Bureau of Standards, U.S. Department of Commerce.

[19] Freedman, M.J., Dingledine, R., Molnar, D., 2001. The free haven project: Distributed anonymous storage service, Lecture Notes in Computer Science, 67–95.

[20] Garfinkel, S., 1994. PGP: Pretty good privacy. O'Reilly and Associates Inc.

[21] Gnutella, 2000. `http://rfc-gnutella.sourceforge.net`.

[22] Gong, L., 2008. JXTA: A network programming environment, Internet Computing, IEEE 5 (3) 88–95.

[23] Govoni, D., Soto, J.C., 2002. JXTA and security, JXTA: Java P2P Programming 251–282. `http://java.sun.com/developer/Books/networking/jxta`

[24] Han, X., Xu, F. et al, 2006. Performance evaluation of JXTA based p2p distributed computing system, CIC '06. 15th International Conference on Computing 391–398.

[25] Huston, G., Ferguson, P., 1998. What is a VPN?. OPENSIG'98 Workshop on Open Signalling for ATM, Internet and Mobile Networks, Toronto

[26] Josefsson, S., 2003. The BASE16, BASE32, and BASE64 data encodings, `http://www.ietf.org/rfc/rfc3548.txt`.

[27] Kaliski, B., 1998. PKCS#7: Cryptographic Message Syntax Version 1.5, `http://www.ietf.org/rfc/rfc2315.txt`

[28] Kaukonen, K., Thayer, R., 1999. A Stream Cipher Encryption Algorithm "Arcfour", Internet-draft, `http://www.mozilla.org/projects/security/pki/nss/draft-kaukonen-cipher-arcfour-03.txt`

[29] Kawulok, L., Zielinski, K. et al, 2004. Trusted group membership service for JXTA, in: Computational Science - ICCS 2004, Lecture Notes in Computer Science Volume 3038.

[30] Li, Z., Dong, Y. et al, 2003 Implementation of secure peer group in peer-to-peer network, in: Communication Technology Proceedings, 2003. ICCT 2003, pp. 192–195.

[31] Zerfiridis, K.G., Karatza H.D., 2004. File distribution using a peer-to-peer network–a simulation study, Journal of Systems and Software, volume 73, 31-44

[32] Mathewson N. Dingledine R. and Syverson P., 2004. Tor: The second generation onion router, Proceeding of the 13th USENIX Security Symposium, 303–320.

[33] Montenegro, C., Castelluccia, G., 2004. Crypto-based identifiers (cbids): Concepts and applications, ACM Trans. Inf. Syst. Secur. 7 (1) 97–127.

[34] NIST, 1995. FIPS PUB 180-1: Secure hash standard, http://www.itl.nist.gov/fipspubs/fip180-1.htm.

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

[36] Simpson, W., 1996. PPP challenge handshake authentication protocol (chap), http://tools.ietf.org/html/rfc1994.

[37] Sit, E., Morris, R., 2002. Security considerations for peer-to-peer distributed hash tables. In: Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS 02), 261 – 269.

[38] SUN Microsystems, JARsigner, http://java.sun.com/j2se/5.0/docs/tooldocs/windows/jarsigner.html.

[39] SUN Microsystems, Keytool, http://java.sun.com/j2se/5.0/docs/tooldocs/windows/keytool.html.

[40] SUN Microsystems, 2001. Project JXTA, https://jxta.dev.java.net/.

[41] SUN Microsystems, 2007. JXTA 2.5 RC1, http://download.java.net/jxta/jxta-jxse/2.5/.

[42] SUN Microsystems, 2007. JXTA v2.0 protocols specification, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[43] Traversat, B., Arora, A. et al., 2003. Project JXTA 2.0 super-peer virtual network.

[44] Wallach, D. S., 2003. A Survey of Peer-to-Peer Security Issues. Software Security - Theories and Systems, Springer Berlin Heidelberg, Volume 2609/2003, 253-258

[45] Vance, A., 2004. SUN's jxta becomes big business play, The Register. http://www.theregister.co.uk/2004/01/30/suns_jxta_becomes_big_business.

[46] Vroonhoven J. V., 2006. Peer to peer security, In Proceedings of the 4th Twente Student Conference on IT,

[47] W3C, 2002. XML-signature syntax and processing, http://www.w3.org/TR/xmldsig-core.

[48] Wahl, M., Howes, T., Kille, S., 1997. Lightweight Directory Access Protocol (v3), http://www.ietf.org/rfc/rfc2251.txt

[49] Wang, Z., Das, S., S.K., Kumar, M., Shen, H., 2007. An efficient update propagation algorithm for P2P systems, Comput. Commun., volume 30, 1106–1115

[50] Yeager, W., Williams, J., 2002. Secure peer-to-peer networking: The JXTA example, IT Professional 4 (2) 53–57.

[51] Yeager, B., 2003. Enterprise strength security on a JXTA P2P network, P2P'03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing.

[52] Yunhao, L., Jinpeng, H. et al, 2005. Access control in peer-to-peer collaborative systems., First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) 835–840.