XML-based security for JXTA core protocols

Joan Arnedo-Moreno

Estudis d'Informàtica, Multimèdia i Telecomunicació,
Universitat Oberta de Catalunya
Rb. Poble nou 156
08018 Barcelona, Spain
E-mail: jarnedo@uoc.edu

Jordi Herrera-Joancomartí

Escola Tècnica Superior d'Enginyeria, Universitat Autònoma de Barcelona
Campus de Bellaterra, Spain
E-mail: jherrera@deic.uab.cat

Abstract: JXTA defines a set of six core protocols specifically suited for ad hoc, pervasive, multi-hop, peer-to-peer (P2P) computing. These protocols allow peers to cooperate and form autonomous peer groups. This paper presents a satisfactory method that provides security services to the core protocols: privacy, authenticity, integrity and non-repudiation. The presented mechanisms are fully distributed and based on a pure peer-to-peer model, not requiring the arbitration of a trusted third party or a previously established trust relationship between peers, which is one of the main challenges under this kind of environments.

Keywords: Peer-to-Peer; Security; Peer group; JXTA; XMLdsig; XMLenc; Peer Resolver Protocol

Reference to this paper should be made as follows: Arnedo-Moreno, J. and Herrera-Joancomartí, J. (xxxx) 'XML-based security for JXTA core protocols', *Int. J. Autonomic Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Joan Arnedo-Moreno is an associate professor at Estudis d'Informàtica, Multimèdia i Telecomunicació in Univeristat Oberta de Catalunya (UOC). He graduated in Computer Science in 2002 at Universitat Politècnica de Catalunya (UPC), where he also got his PhD in 2009. In 2004 he joined UOC in 2004 and joined the KISON research group. His research interests include the topic of security in distributed systems and networking, having published several conference and journal papers.

Jordi Herrera-Joancomart is associate professor at Departament d'Enginyeria de la Informació i les Comunicacions in the Universitat Autònoma de Barcelona. He is graduated in Mathematics by Universitat Autònoma de Barcelona in 1994 and he received his Ph.D. degree in 2000 from Universitat Politècnica de Catalunya. In 2000, he

Copyright © 2009 Inderscience Enterprises Ltd.

joined the UOC and founded the KISON research group in which he is now an external researcher. His research interests include topics in the field of computer security and more precisely in copyright protection techniques and security in ad-hoc networks. He has published more than 60 papers in national and international conferences and journals. He has been main researcher in several national research projects.

1 Introduction

In a peer-to-peer (P2P) network, all involved parties share resources and collaborate in order to provide basic services, such as content, processing or messaging. Under this scenario, it is also assumed (Oram, 2001) that all peers have equivalent capabilities, and a central server with more processing power is no longer necessary. JXTA (SUN Microsystems, 2001) is a set of open core protocols that enable the creation of such networks.

JXTA's core protocols allow peers to cooperate and form autonomous peer groups transparent to their location, as well as to provide the necessary services in order for any JXTA application to operate within the network. Peers may use such protocols in order to advertise and discover resources, to join peer groups and to dynamically route messages across multiple network hops. Furthermore, few assumptions are made about the underlying network transport, in order to guarantee that they may be applied to the broadest set of network scenarios.

As the popularity of JXTA has increased, not only it has become one of the main testbeds for P2P applications, with over 2,700,000 downloads, 120+ active projects and 18,000+ collaborators. Such projects range from simple chat applications (P2PChat, 2008) to systems management (Liberatio, 2006) or real-time collaboration (Connex, 2007) platforms. Furthermore, several companies already have taken advantage of JXTA in order to develop their own products based on a P2P model (Vance, 2004). Some of them are quite important ones, such as Verizon, Nokia or SNSing, China's most used social networking software.

However, since, ultimately, every JXTA application must rely on the use of the core protocols in order to interact with other peers within the network, it is becoming very important to provide capable methods to secure them in order to avoid possible attacks. The current JXTA reference implementation addresses some of these problems, but the provided methods are not fully satisfactory, as they do not fully comply with the JXTA specification ideary of XML data formatting and rely on the existence of a party that must be trusted by all peer group members, reducing the decentralization properties of the system.

The standard security threats to be addressed under a P2P environment can divided into two different groups: passive and active ones (Govoni and Soto, 2002). 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 in search of 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 to achieve 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 routing data alteration.
- *Playback or replay (Pb)*, in which some data exchanged between two legitimate peers is intercepted by the attacker in order to reuse the exact data at a later time to 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.

The contribution of this paper is to provide a modular method to protect JXTA's core protocols against both types of attacks in a manner specifically suited to the protocols' idiosyncracies. Passive attacks are avoided via data privacy whereas active attacks are countered by providing authenticity, integrity and non-repudiation. The presented method does not rely on external parties, keeping a pure P2P model, and authenticity is locally decidable (eliminating the possibility of collusion). This is very important in an environment where messaging may be continuous and other peer's availability is not guaranteed.

This paper is organized as follows. Section 2 provides an overview of JXTA core protocols and the current methods for securing them. Section 3 describes the proposal for improving the current methods, by following the JXTA ideary of XML message formatting. Following, Section 4 provides some insights about the cost of providing a security layer. Concluding the paper, Section 5 summarizes the paper contributions and outlines further work.

2 An Overview of JXTA Core Protocols

Following is a short summary of the six core protocols and its main functionalities (further information can be found in (SUN Microsystems, 2007)):

- 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.
- In order to obtain information regarding to 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)* to send requests to one or several peers and manage responses. The PRP protocol associates a unique ID to every request, which is included in each message. 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, acting as abstract endpoints above any physical network.
- Routes between peers are found with the help of the *Endpoint Routing Protocol (ERP)*. Whenever a peer is about to send a message to a destination ,but does not know the path, an ERP message is sent to other peers to locate an existing 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.

It is not mandatory for JXTA implementations to deploy all core services, but at least PDP and ERP must be supported in order to provide addresses to peers and allow communication between endpoints. The remaining protocols are optional, but supporting them increases interoperability and provides a wider degree of functionality. The current JXTA J2SE implementation (SUN Microsystems, 2001) supports all of them.

In order to accommodate into the P2P model, all protocols are asynchronous, best effort and based on a query-response model. Whenever a query is sent to other peer group members, zero, one, or more responses may be received.

As it is shown in Figure 1, JXTA endpoint communication is structured in a classical layered approach, the core protocols acting as a gateway to networking operations under a P2P environment. This provides an abstraction layer to both JXTA's own services and custom made application dependant ones (operating at the upper Services layer), enabling the deployment of services in a transparent manner to the real underlying transport methods or topology.

At the Peer layer, the higher level core protocols (PIP, RVP, PBP and PDP) allow services to publish, locate and exchange resources. The Endpoint layer manages routing and addressing, via the ERP protocol, and specifies the format for all query-response exchanges, using the PRP protocol. This means that all core protocols' queries sent across the network are ultimately encapsulated into a PRP query. PRP queries are then encapsulated as messages at the Messaging layer. Finally, the message is sent across the network using any of the available wire transport protocols (chosen according to the application's needs) at the Wire Transport layer, such as TCP, HTTP, multicast or TLS. The message generated at the Messaging layer is considered the application level data to be sent by the wire transport protocol.

JXTA structures messages as a set of named and typed content elements, which means that it is essentially a set of name/value pairs, organized as an ordered sequence. The content element can be any arbitrary type and the most recently added element is always appended at the end of the message. As a message passes down each layer, one or more named elements may be added to the message. As a message passes back up the stack, each layer will remove these elements.

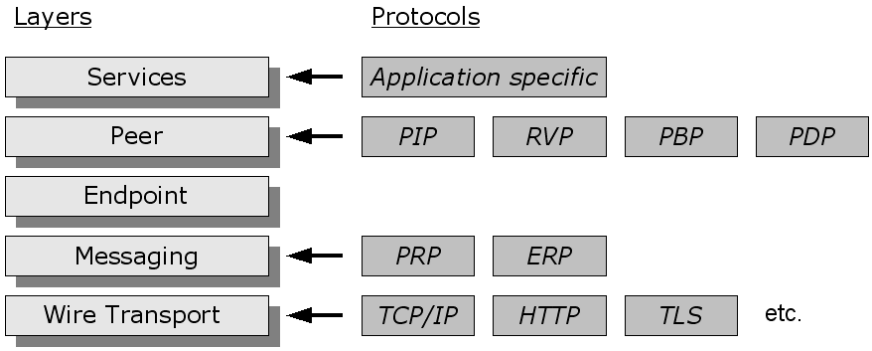

Figure 1 JXTA protocol layers.

All core protocols are codified using XML, each message element name being the root XML element tag and its content the corresponding XML subtree. The main reasons for such format are its programming language/platform independence, being self-describing and being able to enforce correct syntax. As an additional feature, XML can easily be translated into other encodings, such as HMTL, which may allow peers that do not support XML to access resources. The XML schema for a PRP query is shown in Listing 1 as a sample of core protocol format. It is an interesting election, since all Peer layer protocols use PRP to transmit messages (by encapsulating its own content into the `Query` element).

XML Listing 1 - PRP Query schema

```
<xs:complexType name="ResolverQuery">
  <xs:sequence>
    <xs:element ref="jxta:Cred" monOccurs="0"/>
    <xs:element name="SrcPeerID" type="jxta:JXTAID"/>
    <xs:element name="HandlerName" type="xs:string"/>
    <xs:element name="QueryID" type="xs:string"/>
    <xs:element name="HC" type="xs:unsignedInt"/>
    <xs:element name="Query" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

2.1 Current security in JXTA's core protocol messaging

Since any JXTA service relies on the underlying core protocols in order to send messages across the network, they become one of the building blocks of the JXTA architecture. Unless security measures are available, it will be easy to subvert the network by attacking peer communications. In this section, we provide a brief overview of the current security capabilities of JXTA in messaging. A much more complete survey on JXTA security may be found in (Arnedo-Moreno and Herrera-Joancomartí, 2009).

In the current reference implementation, messaging has been secured under the assumption that the Personal Security Environment (PSE) has been chosen

as a group's Membership Service. The Membership Service is one of the JXTA core services, taking care of group membership and identity management within a peer group by providing each group member with a credential. Peers may include credentials in messages exchanged within a peer group in order to prove group membership and provide a means for implementing access control in offered services.

How this credential is claimed depends on the type of Membership Service being used within a peer group. The JXTA reference implementation, as far as version 2.5, provides three basic Membership Services: None, Password and PSE. However, the JXTA specification lets the door open to new types of custom Membership Services.

The PSE Membership Service is the default one in JXTA when a secure environment is needed, its credentials being based on PKIX (CCITT, 1988) certificate chains. The group creator holds the root certificate, acting as a certification authority. An identity is claimed by being able to access the keystore entry which holds the private key for that certificate chain. Since PSE is based on public key cryptography, its credentials are chosen as a means to provide asymmetric key management for messaging security services.

By using the PSE membership Service, JXTA messages may be secured at two different layers: at the messaging layer, by using the CBJX (Bailly, 2002) protocol, and at the wire transport layer, via its own definition of TLS (Dierks and Allen, 1999).

2.1.1 Messaging layer security

The standard messaging layer 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 its own self-defined digital signature element into messages, providing data integrity and authentication. This approach provides protection against active threats.

Even though CBJX is specified as a wire transport protocol, it can be truly considered to operate at the messaging layer (or, more exactly, at a meta-messaging layer). The main reason is that it lacks the capability to directly send messages between endpoints, which is what actually defines a wire transport protocol in JXTA. CBJX pre-processes messages 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 layer security.

In addition to the original message's digital signature, an information block, according to the XML schema definition shown in Listing 2, is also encapsulated with the secured message: a `CbJxMessageInfo` XML data structure, 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.

XML Listing 2 - 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>
```

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 2. CBJX encapsulates signatures by using a single `Signature` XML element containing a Base64-encoded PKCS#7 (Kaliski, 1998) binary signature.

On reception, the CBJX information block is unencapsulated and both signatures are validated, invisibly acting as far as upper layer protocols is concerned, by providing the original message.

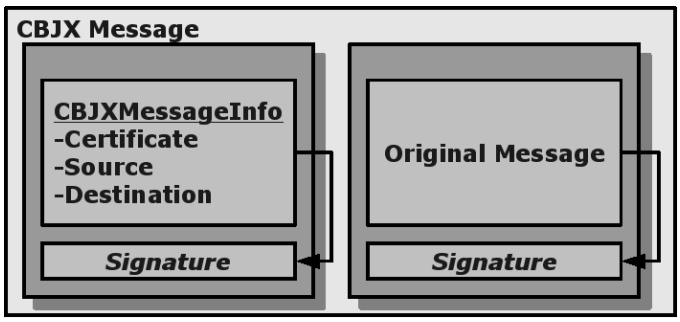

Figure 2 CBJX secure encapsulation.

Apart from digital signatures, CBJX provides an additional lightweight authenticity method by using Crypto-Based Identifiers (CBIDs (Montenegro and Castelluccia, 2004)), which will be discussed in Section 3. At this point, only notice that this method provides authentic messaging without the need of certificates issued by a TTP (Trusted Third Party). Self-signed certificates are good enough.

2.1.2 Wire transport layer security

The JXTA definition of standard TLS provides private, mutually authenticated, reliable streaming communications. Thus, TLS provides protection 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 encryption (e.g., DES (FIPS, 1977), RC4 (Kukonen and Thayer, 1998), 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 prior to any data exchange. 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 encapsulate 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.

3 A Modular XML-based Security Approach

From the explanation in Section 2, it can be concluded that the current security layer in core protocol messaging has three main shortcomings which could be improved:

- Lose of interoperability: Even though JXTA makes heavy use of XML in its protocols, no XML standard is used for signature generation at the messaging layer (namely, in CBJX), which may become a hurdle if interoperability is to be maintained between applications. Signatures are generated by serializing XML as plain text, which is not a good solution for XML processing, as will be explained in Section 3.2.

  Furthermore, under the current methods (both TLS and CBJX) endpoints must support and agree to use the provided security layer in order to communicate. It is not possible for a peer which chooses not to deploy the security layer (for example, because of computational limitations) to understand received secure messages.

- No privacy at the messaging layer: Currently, the only way to achieve data privacy, in order to counteract passive attacks such as eavesdropping or traffic analysis, is using TLS. However, since TLS is a wire transport protocol, it imposes a constraint that cannot be ignored: no other transport protocol may be used underneath. That means that it is not possible to support JXTA's message propagation via multicast, as well as plain HTTP proxies either. Furthermore, as new transport protocols become supported in JXTA (for example, UDP or RTP), it will not be possible to use them in a secure manner.

- TTP-based trust model: The use of CBJX or TLS forces peer group management via the PSE Membership Service. PSE provides an integrated secure environment in JXTA, but for some applications it may become too restrictive by constraining the peer group trust model to one based on a TTP, and forcing the use of X509 certificates. This is not always desirable in a dynamic and decentralized environment such as P2P specially when trying to maximize peer equality and self-organization. In addition, the use of a TTP inherits additional problems which increase the system's complexity, such as that of certificate chain management and revocation.

In order to solve this issues, we move a step further from the security proposal in (Arnedo-Moreno and Herrera-Joancomartí, 2008) to deploy security at the Messaging layer, which is common to all of JXTA's core protocols. This proposal also takes advantage of the same CBID format and secure key distribution as presented in (Arnedo-Moreno and Herrera-Joancomartí, 2008) to guarantee public key authenticity. It must be noted that advertisement security in that proposal is not overridden by this approach. Both methods nicely complement each other, since it may still be necessary to apply persistent security to advertisements when finally delivered to the Services layer and stored into the peer's local cache.

We define a data encryption and signature format for PRP messages based on XML standards, making use of the standard JXTA messaging signature capabilities as explained in subsection 2.1.1. Using this method, the message format, as defined in the JXTA v2.0 protocols specification (SUN Microsystems, 2007), is maintained. Therefore, peers which do not support signatures may nevertheless invisibly process messages. As a result, each peer may choose its own degree of security, at its own risk, without being constrained by other peer's decision on that regard.

3.1 Core protocol privacy

Core protocol privacy is achieved by encrypting the XML message content. There are several approaches to achieve selective encryption in XML documents (Hada and Kudo, 2002; Geuer-Pollmann, 2002; W3C, 2002a). For our proposal, we specifically use *XMLenc* (W3C, 2002a), since in conjunction with its complementary standard *XMLdisg* (W3C, 2002b), they both provide a full security set for passive and active attacks against XML data. Since all core protocols in JXTA are XML-formatted, it is a logical choice. Additional advantages are its status as an XML standard, its flexibility and its capability to guarantee data privacy during transit or when stored in parties different from the one which

generated the document, which is not supported in (Hada and Kudo, 2002) and providing a reasonable result document size, in contrast with (Geuer-Pollmann, 2002).

Messages are selectively encrypted using a wrapped key encryption scheme (such as the one defined in (Kaliski and Staddon, 1998)). For each message field to be encrypted, a symmetric key is generated and used to encrypt the field. The symmetric key is then encrypted (wrapped) using each recipient's public key, obtaining a set of encrypted keys, each one of which can only be accessed by only one of the recipients.

This scheme is applied to PRP messages according to the profile shown in Figure 3. Wrapped keys are included into the encrypted message by encapsulating the original message into an `EncryptedMessage` XML element and introducing an additional XML element as a sibling, the `KeyList` element. For each encrypted field, a set of wrapped keys is included in the `KeyList` element. This relationship is shown as arrows in the figure.

Under this scheme, it is not mandatory to provide an all or-nothing approach, the sender may choose which message fields will be encrypted and which will be left as plain text.

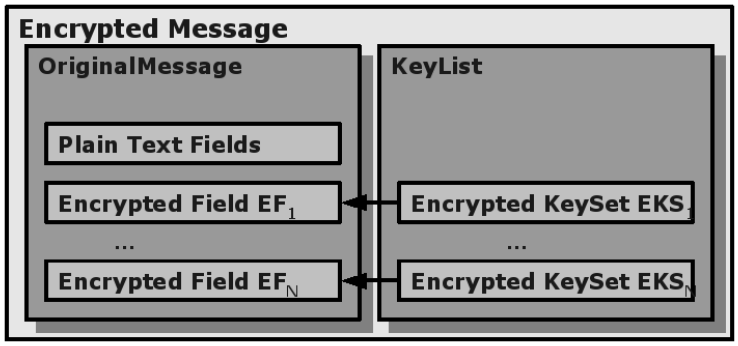

Figure 3 Message encryption profile.

Each wrapped key within a keyset is defined by an XMLenc `EncryptedKey` element and contains all the cryptographic information necessary to decrypt such field. An `EncryptedKey` element exists for each peer which may access the encrypted field

Encrypted fields within a message are defined by XMLenc `EncryptedData` elements. Figure 4 shows how each `EncryptedData` element is linked to its corresponding `EncryptedKey` elements. Each peer's message security layer identifies which `EncryptedKey` fields may be decrypted with the local peer's private key by searching for its Peer ID in the `KeyInfo` field of each contained `EncryptedKey` element. All `CarriedKeyName` subelements within the same key set always point to the same `EncryptedData` element.

By using this XML profile and message schematics, it is possible to accommodate selective field encryption. Since the final encrypted data is sent to

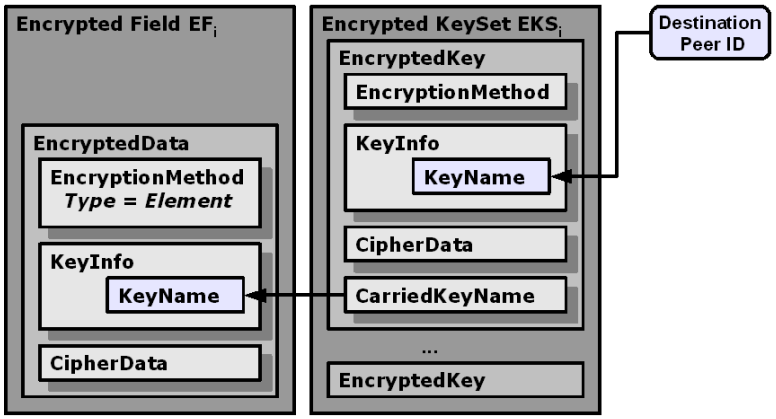

Figure 4 XMLenc encryption profile.

wire transport protocols as a standard JXTA message, encrypted fields become invisible to its basic operation.

3.1.1 Encryption and decryption process

The process of message encryption that generates the XMLenc profile previously defined can be described as follows.

Encryption:

1. Each member of the peer group distributes its public key as will be explained in Section 3.3.
2. Peer A needs to send a core protocol message.
3. A new `KeyList` element is generated.
4. For each element in the message, peer A chooses the subset of peers P_j, for $j = 1, \cdots, m$, which will be able to access it.
5. Each element field F_i is encrypted in the following manner:
   (a) Both a random symmetric key k_i and an identifier id_i are generated by A. A new `KeySet` element is generated.
   (b) F_i is encrypted according to *XMLenc* with k_i. The original field becomes an XMLenc `EncryptedData` element.
   (c) For each peer P_j, for $j = 1, \cdots, m$:
      i. A retrieves PK_j, the public key of P_j.
      ii. k_i is wrapped (encrypted) using PK_j, generating an XMLenc `EncryptedKey` element. The `CarriedKeyName` field of such element is set to id_i. Its `KeyInfo` field is set to P_j's Peer ID by using a `KeyName` element as previously shown in Figure 4.

iii. The `EncryptedKey` element is added to the `KeySet` element.

(d) Once all peers in P_j, for $j = 1, \cdots, m$, have been processed, the `KeySet` element includes the wrapped keys for all peers P_j, for $j = 1, \cdots, m$. That means, all peer which may access the field.

(e) The newly generated `KeySet` element is appended to the current `KeyList` element.

6. An encrypted message has been generated according to the format previsouly defined. For each encrypted field F_i, a set of wrapped keys (a `KeySet` element) exist within the `KeyList` element which may decrypt it.

7. The message is sent via the chosen wire transport protocol.

A sample encrypted message after this process is shown in Listing 3 (some ID's and Base64 encoded data have been shortened in order to improve readability). This message contains a PRP query (the XML schema for such queries was shown in Figure 1). Only the original `Query` element has been encrypted, and two recipients (with Peer ID *urn:jxta:uuid-59...C03* and *urn:jxta:uuid-59...F03*) may properly decrypt it, as the existence of two `EncryptedKey` elements shows.

Decryption:
Whenever a peer $B = P_j$ for some $j = 1, \cdots, m$ wants to access to the resource:

1. Peer B's Messaging layer receives an encrypted message.

2. B locates the `KeyList` element.

3. B locates within the `KeyList` element the set of `EncryptedData` elements, ED, which contain B's Peer ID in its `KeyInfo` field.

4. For each `EncryptedData`, Enc_i, in ED:

   (a) The field to be processed, F_i, is located by matching its `KeyName` field value with Enc_i's `CarriedKeyName` field value, which must be equal.

   (b) B's private key is used to decrypt the symmetric key, k_i, stored in Enc_i .

   (c) The original F_i is recovered by decrypting it using k_i.

5. B obtains the original message where some elements may still be encrypted. B has no access to such entries, only to those that could be satisfactorily decrypted.

3.2 Core protocol authenticity and integrity

Authenticity and data integrity (as well as non-repudiation) is provided to core protocols by using XML signature (XMLdsig) at the messaging layer. The process is similar to the method proposed in (Arnedo-Moreno and Herrera-Joancomartí, 2008) for advertisement authenticity, although it has been specifically adapted for core protocol messaging.

XML Listing 3 - Selectively encrypted message

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlsecure:EncryptedMessage>
  <jxta:ORes>
    <ResolverQuery>
      <HandlerName>urn:jxta:uuid-DEADBEEF...05</HandlerName>
      <QueryID>0</QueryID>
      <HC>0</HC>
      <SrcPeerID>urn:jxta:uuid-59...503</SrcPeerID>
      <xenc:EncryptedData xmlns:xenc="...xmlenc#"
          Type="...xmlenc#Element">
        <xenc:EncryptionMethod
             Algorithm="...xmlenc#aes128-cbc"/>
        <ds:KeyInfo xmlns:ds="...xmldsig#">
          <ds:KeyName>My Encrypted Data</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
           <xenc:CipherValue>Pr...It/4</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedData>
    </ResolverQuery>
  </jxta:ORes>
  <KeyList>
    <KeySet>
      <xenc:EncryptedKey xmlns:xenc="...xmlenc#">
        <xenc:EncryptionMethod Algorithm="...xmlenc#rsa-1_5">
        <ds:KeyInfo xmlns:ds='...xmldsig#'>
          <ds:KeyName>urn:jxta:uuid-59...C03</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>N9...9w==</xenc:CipherValue>
        </xenc:CipherData>
        <CarriedKeyName>My Encrypted Data</CarriedKeyName>
      </xenc:EncryptedKey>
      <xenc:EncryptedKey xmlns:xenc="...xmlenc#">
        <xenc:EncryptionMethod Algorithm="...xmlenc#rsa-1_5">
        <ds:KeyInfo xmlns:ds='...xmldsig#'>
          <ds:KeyName>urn:jxta:uuid-59...F03</ds:KeyName>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>A3...7f==</xenc:CipherValue>
        </xenc:CipherData>
        <CarriedKeyName>My Encrypted Data</CarriedKeyName>
      </xenc:EncryptedKey>
    </KeySet>
  </KeyList>
</xmlsecure:EncryptedMessage>
```

Apart from keeping message readability, XMLdsig offers some capabilities which are important in this environment.

First of all, it maintains interoperability by taking into account XML canonicalization (Boyer, 2001). This is extremely important when using XML, since documents which are syntactically different may translate as semantically equal (for example, changing order of sibling XML elements). Directly feeding XML data to a signing algorithm, as is exactly the case for CBJX, does not take this fact into consideration, since a single different bit, however irrelevant to the XML semantics, will invalidate a signature. This is not an improbable occurrence in an heterogeneous network, where different peers may be using different XML parsers (for example, simply because they are running different operating systems). Just for that only reason, using XMLdsig becomes very important when signing XML data.

Finally, XMLdsig is an open specification which allows the definition and inclusion of new types of credentials in order to transport the public keys which validate the signature. This advantage allows to support a wide variety of standard credentials, instead of being constrained to only PKIX certificates. In addition, it is ready to support any new type of credential which JXTA or any specific application decides to use, by just assigning a new URI type to the XMLdsig key transport element (the `KeyInfo` element).

In this proposal, a detached signature is used within the message body, as shown in Figure 5. The XML signature is included as a message signature, just as is the case in CBJX, but instead of a self-defined single `Signature` element, a full XMLdsig signature is included. In contrast with CBJX, no additional encapsulation is needed, since the XML signature contains all needed information related to the security layer. As a result, messages generated using this method may be processed even by peers which do not support signatures (they are able to decode the original message, nevertheless, by just ignoring the signature).

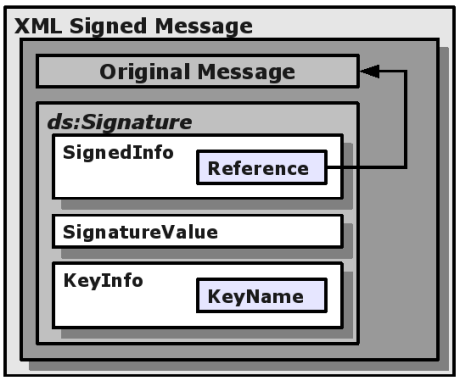

Figure 5 XMLsig detached signature profile.

Since a detached signature is used, a default URI in the `Reference` element is enough for the Messaging layer to locate the corresponding signed data (the original message). The `KeyName` element is used to retrieve the signer's public key in signature validation, as will be shortly explained in Section 3.3.

3.2.1 Signature processing

The signature process is straightforward, since the Messaging layer at the signing peer holds all the required information: the peer's private key and the message to be signed.

In order to validate a signed message, the following steps are necessary:

1. Retrieve the source peer's public key.
2. Apply the SHA-1 hash algorithm (NIST, 1995) to the public key to generate a JXTA CBID.

3. Compare the resulting CBID with the source peer CBID. If equal, key authenticity is proved.

4. Validate XML signature using the public key retrieved in Step 1. If valid, integrity, authenticity and non-repudiation are proved.

See (Arnedo-Moreno and Herrera-Joancomartí, 2008) for the details about the CBID is generated in steps 1 and 2. Key retrieval will be discussed in Section 3.3.

A sample message signature is shown in Listing 4 (some ID's and Base64 encoded data have been shortened). Notice that, since it is a detached signature, it is not necessary that the message is explicitly present. It is possible to realize that it is a detached signature since an implicit URI is used at the `Reference` element.

XML Listing 4 - Signed message (detached signature)

```
<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>
      <ds:Transforms>
        <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>KoOR31wMpcJ17VAmtaUf7nS/KU4=
      </ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue> nloCRUg4WBOH+DcEAuLKGYhqvsfdRCy4R...
    ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
  </ds:SignatureValue>
  <ds:KeyInfo>
    <KeyName>urn:jxta:uuid-596162...BCF0646C103</KeyName>
  </ds:KeyInfo>
</ds:Signature>
```

Both XMLenc and XMLdsig validation nicely integrate since XML signature processing automatically detects that some signed data is encrypted, so it must be previously decrypted before signature validation. This is achieved by including the appropriate XMLenc `Transform` element within the signature `Reference` element.

3.3 Public key distribution and retrieval

The proposed security mechanisms (encryption and signature) imply that each peer must distribute its public key across the peer group, so other group members may send encrypted data as well as validate signatures. To achieve this purpose, we take advantage of JXTA's resource publication methods by using each peer's

Peer Advertisement for key transportation. Peer Advertisements also act as a presence mechanism, announcing every peer's basic information to other group members, and are routinely exchanged at fixed time intervals during JXTA's standard operations. Using this approach, no extra message types or additional protocols are necessary.

JXTA manages advertisement publication and retrieval using the Shared-Resource Distributed Index (SRDI) (Abdelaziz et al., 2003) network, formed by Rendezvous Peers (JXTA super-peers). Rendezvous Peers act as a remote advertisement index cache. The SRDI network is accessed using the Discovery Service, one of JXTA's core services, which relies on PDP messages to push and locate resource metadata, such as advertisements. The SRDI network also allows peers outside the local network (out of broadcast range) to retrieve advertisements and as well as enabling group members which where off-line for some time to retrieve advertisements published during their disconnection. Whenever a peer receives the advertisement, it is automatically indexed, stored in a local cache and assigned an expiration date.

The key publication process using this method can be summarized in the following steps:

1. At JXTA boot time, the peer creates its Peer Advertisement.
2. The advertisement is signed using the approach proposed in (Arnedo-Moreno and Herrera-Joancomartí, 2008). The peer's public key is included in the signature's `KeyInfo` element.
3. The advertisement, which now contains the peer's public key, is stored and indexed in the peer's local cache, as JXTA's standard operation dictates.
4. At fixed time intervals, the Peer Advertisement index, which, among other data, contains the peer's ID, is pushed to a Rendezvous Peer using JXTA's Discovery Service.
5. The Peer Advertisement index is distributed and replicated across the SRDI network.
6. The public key is then considered distributed across the peer group.

The public key of a particular peer can be retrieved by querying the SRDI network. An outline of such process follows:

1. Using the Discovery Service, the requesting peer, RP, queries any Rendezvous Peer for the key owner's, KO, Peer Advertisement. KO's ID is used to unambiguously identify the specific advertisement being requested.
2. The query is distributed across the SRDI network, until the target advertisement's index is found, so the advertisement owner, KO can be located.
3. The query is then sent to KO.
4. KO pushes the advertisement to RP, using the PDP protocol.

5. The advertisement's signature is validated.

6. If the signature and CBID are valid, the requested public key can be found in the `KeyInfo` XML element. Otherwise, it is deemed unauthentic and discarded.

7. The advertisement is stored in *RP*'s local cache. The next time *KO*'s public key is required, it can be directly obtained from the local cache, without querying the SRDI network, speeding up the process.

Figure 6 shows the schematics of steps 1-4, advertisement retrieval.

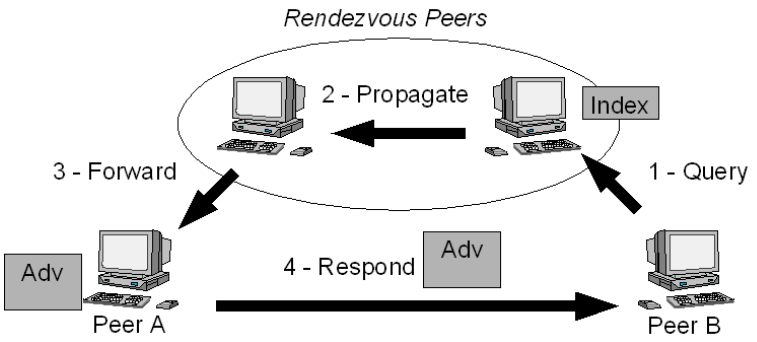

Figure 6 Advertisement publication and retrieval.

Using JXTA's SRDI as a method for key transport also takes into consideration some important characteristics of the a JXTA network. First of all, those peers which decide not to support messaging security are still able to recognize Peer Advertisements as such, just ignoring the additional key transport fields. Therefore, key distribution can also properly operate in heterogeneous networks, where each peer is free to chose it own security level. Finally, this method also guarantees that the destination peer public key is always locally available prior to a message exchange, since in JXTA it is not possible to communicate with another peer without previously locating its Peer Advertisement.

In signed PRP messages, in order to retrieve the Peer Advertisement (and the corresponding public key), the source Peer ID can be obtained from the contained `KeyName` element in the signature. In the case of encrypted PRP messages, it is obvious that the sender must already know the destination Peer ID. With that ID, it is possible to find the corresponding Peer Advertisement querying the SRDI service.

A sample of the new Peer Advertisement format that enables public key distribution for a particular Peer ID is shown in Listing 5. An XML signature is used to transport the public key. However, in this case an enveloped approach is used, instead of a detached one, and a raw RSA public key is being distributed. The advertisement is also signed in order to ensure key authenticity and data integrity.

XML Listing 5 - Key distribution via Peer Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PA>
<jxta:PA xml:space="preserve" xmlns:jxta="http://jxta.org">
 <PID>urn:jxta:uuid-59...B03</PID>
 <GID>urn:jxta:jxta-NetGroup</GID>
 <Name>My Peer Advertisement Name</Name>
 <Desc>My Peer Advertisement Description</Desc>
 <Svc>
   <MCID>urn:jxta:uuid-DE...05</MCID>
   <Parm>
   <jxta:RA><Dst><jxta:APA>
     <EA>tcp://...:9701</EA>
     <EA>cbjx://uuid-59...03</EA>
     <EA>jxtatls://uuid-59...03</EA>
     <EA>relay://uuid-59...03</EA>
   </jxta:APA></Dst></jxta:RA>
   </Parm>
 </Svc>
 <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> nloCRUg4WBOH+DcEAuLKGYhqvsfdRCy4R...
     ...QYH8Czizo3P AkvLI1UGoMekOHRL2kI=
   </ds:SignatureValue>
   <ds:KeyInfo>
     <ds:KeyValue><ds:RSAKeyValue>
       <ds:Modulus>mHkQ5C6WU=</ds:Modulus>
       <ds:Exponent>AQAB</ds:Exponent>
     </ds:RSAKeyValue></ds:KeyValue>
   </ds:KeyInfo>
 </ds:Signature>
</jxta:PA>
```

Key authenticity is ensured by signing advertisements and using CBIDs as JXTA Peer IDs, which also ensures that active attacks do not subvert the original data. Signatures are generated using an encapsulated XMLdsig signature type. Its `KeyInfo` element is used for key transport, however, several key types are supported by using different subelement types: `KeyValue` (raw public key, used in the example), `X509Data` (X509 certificate), `PGPData` (PGP (Garfinkel, 1994) key) and `SPKIData` (SPKI (Ellison, 1999) certificates).In addition, the XMLdsig standard supports the definition of new key transport types.

4 Cost evaluation

The proposed security solution for JXTA protocols implies a cost in message size. Some overhead has to be accepted, since cryptographic information must be added to the original content. In this section we study the resulting overhead for the proposed XML encryption and signature profiles.

In order to provide some typical values for message and advertisement sizes, we have based our study in the JXTA framework and applications provided by the JXTA-Overlay project (Xhafa et al, 2009). We consider it is an interesting choice, since this project was conceived so it would be able to adapt to a broad set of P2P applications. Furthermore, security was not considered at all during its design stage, so it is vulnerable to all threats introduced in Section 1. Therefore, all our assumptions in this section will be based on JXTA-Overlay's operation.

4.1 XML encryption cost

In the case of achieving privacy via XML encrytpion, the cost in additional message length comes from two different element types (see Listing 3): `EncryptedKey` and `EncryptedData`. We must take into account that the former is a new addition to the unsecured message, whereas the latter just substitutes the original field content.

The `EncryptedKey` element size is quite static, its size only depends on the algorithm identifier, in the `EncryptionMethod` field, and the chosen `CarriedKeyName` identifier. For an RSA 1.5 algorithm, the default one for public key encryption in XMLenc, with 1024 bits key length and a 6 bytes long identifier, its size is always about 570 bytes. The length of the chosen symmetric key has no impact on this value if it is under the RSA modulus.

In contrast, the `EncryptedData` size varies, depending on the original unencrypted field size, $size_{of}$. Also assuming a 6 bytes long key identifier, for an AES symmetric encryption algorithm the total size is about $403 + \frac{4}{3} size_{of}$.

Taking these considerations into account, the final cost in additional message length will be highly dynamic depending on several factors: the number of encrypted fields, the size of the original data for each individual encrypted field and the number of destination peers. Since there are many factors to take into account, it is difficult to provide an overhead function without making many additional assumptions, such as the number of unencrypted fields and their individual size. For example, the main content field size of a typical PDP query is about 450 bytes long, whereas in the case of an ERP response it is about 1480 bytes long. Furthermore, since JXTA does not provide any privacy mechanism at message level, it is not possible to provide any direct comparison with existing methods.

Nevertheless, Figure 7 shows the estimated cost for some values on regards to number of destination peers and encrypted fields, assuming an average field size of 850 bytes. The values in the figure measure the increase in size in additional bytes because of encryption. This value has been chosen since it is the typical size of a PDP response field (a resource publication message), which is the most common one during the peer life cycle as far a PRP messages is concerned, and can be considered sensitive data that should be protected.

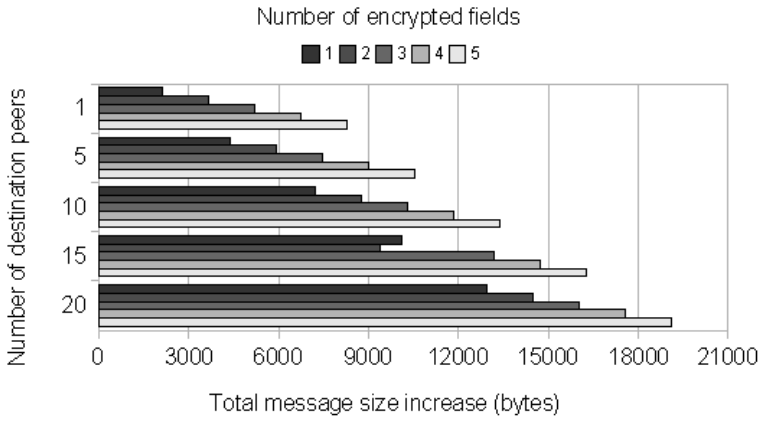

Figure 7 Signed PRP message size increase

4.2 XML signature cost

The overhead in the proposed message integrity and authenticity method comes from the `Signature` element, which is a completely new addition to the original unsecured message content. There are no element replacements, as was the case with XML encryption. Furthermore, once the key transport method is set, the size of a `Signature` element is the same for all messages, remaining static. Original content size has no impact on signature size.

In contrast with encryption, JXTA provides CBJX as a message level mechanism to achieve data integrity and authenticity. Therefore, it is possible to compare our proposal's overhead with CBJX's one. This comparison is shown in Figure 8. By analyzing the PRP message XML schema, we have considered that a good estimation for size reference is about 1Kb for small sized ones, 3Kb for medium sized ones and 5Kb for the bigger ones.

Since XMLdsig supports several key transport element types, the two most significative ones are included: `X509Data` and `KeyName`. In the former, the signer's X509 certificate is explicitly included into the signature, whereas in the latter only an identifier is included. Our key distribution method assumes a `KeyName` key transport, even though it fully supports a `X509Data` one.

In all cases, overhead is quite high for small messages but quickly falls as message size increases (generally speaking, the size of secure encapsulation is about 2Kb). However, from our analysis, we can see that using a `KeyName` key transport, our proposal's overhead is better that CBJX, specially for small messages. In the case of an `X509Data` transport, which directly includes the signer's certificates, just as CBJX does, the overhead is not much higher.

In our proposal, some extra overhead is incurred at other protocols apart from PRP, since each peer's certificate must be distributed (using the method exposed in subsection 3.3). However, CBJX always operates under a PSE Membership Service, which also distributes each peer's certificate, and, in addition, the TTP's root certificate. Therefore, CBJX's impact on other JXTA protocols is higher, since even more data must be distributed to all peer group members. Since the

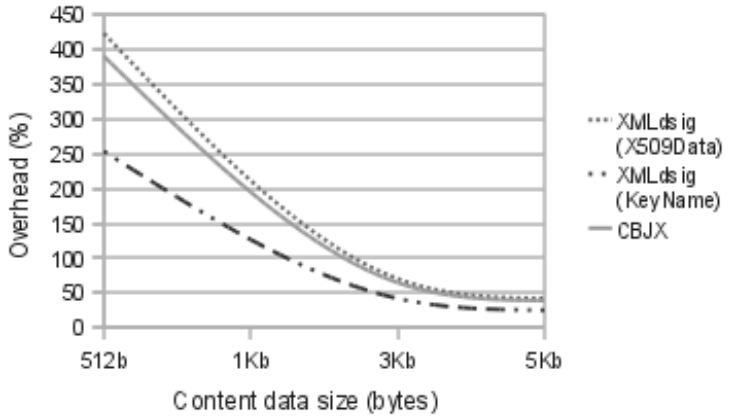

Figure 8 Signed PRP message size overhead comparison.

size of key distribution messages are quite static, it is feasible to compare direct numbers regarding the overhead of our proposal against CBJX, using JXTA-Overlay's standard operation as a basic framework.

The amount of data that must be transmitted in order to securely distribute a key, so CBJX may properly operate amounts to 4902 bytes (2971 bytes for peer certificate itself and 1931 for the TTP certificate). This information is updated every 10 seconds. In contrast, our proposal only needs to transmit 1824, also every 10 seconds. Therefore, there's an improvement of about 62%.

5 Conclusions and Further Work

A new proposal for core protocol messaging security in JXTA has been presented. Its main contributions are threefold.

First of all, the proposed method provides two flavors of security services in order to thwart network threats: on one side, data privacy, and on the other side, authenticity, integrity and non-repudiation. As a result, both passive and active threats are taken into account (in contrast to CBJX, which only subverts active attacks). They are specified in a modular way and without the need of a TTP, which is important in P2P, as well as not being constrained to a specific Membership Service o credential type (PKIX certificates). Services and applications may choose which flavor of security is most convenient and apply only the necessary one (or both).

In addition, by deploying protocol security at the messaging layer, it is now possible to provide data privacy using any wire transport protocol. In this manner, applications are not bound to a specific one, and can choose which to use according to their needs across the full range. As a result, it is now possible to provide privacy to JXTA's message propagation transport protocols as well as using an enabling asynchronous messaging, in contrast with TLS, which requires synchronous negotiation over a reliable channel prior to data exchanges.

Finally, our proposal keeps a high degree of interoperability by using the standard messaging signature element inclusion capability, but via a XMLdsig and XMLenc standards. By applying security at the messaging layer, communication is possible between peers which choose to apply security and those that do not. By using XMLdsig, it is ensured that signatures will be valid in heterogeneous networks, something that current approaches may not guarantee. This is extremely important in a P2P environment.

Further work goes toward using the provided non-repudiation mechanisms to define an incrimination protocol to inform to group members that a specific peer has been compromised or gone rogue. Using collaboration mechanisms, such peer should be isolated and expelled from the group, its traffic being ignored.

Acknowledgements

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.

References

Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J.C., Pouyoul, E., Yeager, B., Traversat, B., Arora, A. (2003). "Project JXTA 2.0 super-peer virtual network", Tech. Rep., Sun Microsystems, Inc.

Arnedo-Moreno, J. and Herrera-Joancomartí, J. (2008). Persistent interoperable security for JXTA. In *Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'08)*, pages 354–359. IEEEPress.

Arnedo-Moreno, J. and Herrera-Joancomartí, J. (2009). A survey on security in JXTA applications. *Journal of Systems and Software*, doi:10.1016/j.jss.2009.04.037, Elsevier.

Kaliski, B. and Staddon, J. (1998). PKCS#1: RSA cryptography specifications. version 2.0.

Bailly, D. (2002). CBJX: Crypto-based JXTA (an internship report). Tech. Rep. Sun Microsystems Inc.

Boyer, J. (2001). Canonical XML. version 1.0. http://www.w3.org/TR/xml-c14n.

CCITT (1988). Recommendation X .509, The Directory - Authentication Framework.

Connex Project Homepage (2007), http://connex.sourceforge.net.

Dierks, T. and Allen, C. (1999). IETF RFC 2246: The TLS protocol version 1.0. http://www.ietf.org/rfc/rfc2246.txt.

Eastlake, C. (2001). US Secure Hash Algorithm 1 (SHA1). http://www.ietf.org/rfc/rfc3174.txt

Ellison, C .Frantz, B. *et al* (1999). SPKI certificate theory. http://www.ietf.org/rfc/rfc2693.txt

FIPS - Federal Information Processing Standard (1977). Data Encryption Standard. National Bureau of Standards, U.S. Department of Commerce.

Garfinkel, S. (1994). *PGP: Pretty Good Privacy.* O'Reilly and Associates Inc.

Geuer-Pollmann, C. (2002). XML pool encryption. In Proceedings of the 2002 ACM workshop on XML security (XMLSEC'02). pp 1-9. ACM

Govoni, D., Soto, J.C. *et al* (2002). JXTA and security. *JXTA: Java P2P Programming*, pages 251–282. Sams Publishing

Hada, S. and Kudo, M. (2002). XML Access Control Language: Provisional authorization for XML documents. `http://www.research.ibm.com/trl/projects/xml/xss4j/docs/xacl-spec.html`.

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

Kaukonen, R., Thayer, R. (1999). A Stream Cipher Encryption Algorithm (Arcfour). `http://www.mozilla.org/projects/security/pki/nss/draft-kaukonen-cipher-arcfour-03.txt`.

Ohlolh Inc. (2006), Liberatio Systems Management, `http://www.ohloh.net/p/liberatio`.

Montenegro, G. and Castelluccia, C. (2004). Crypto-based identifiers (CBIDs): Concepts and applications. *ACM Transactions on Information Systems Security*, Vol. 7, No. 1, pp. 97–127. ACM

NIST (1995). FIPS PUB 180-1: Secure Hash Standard `http://www.itl.nist.gov/fipspubs/fip180-1.htm`.

Oram, A. (2001). *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc.

P2PChat Project Homepage (2008), `https://p2pchat.dev.java.net`.

Sun Microsystems Inc. (2001). Project JXTA. `http://www.jxta.org`.

Sun Microsystems Inc. (2001). JXTA JXSE Programmer's Guide 2.5. `https://jxta-guide.dev.java.net`.

SUN Microsystems Inc. (2007). JXTA v2.0 protocols specification. `https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html`.

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`.

W3C (2002a). XML encryption syntax and processing. `http://www.w3.org/TR/xmlenc-core`.

W3C (2002b). XML signature syntax and processing. `http://www.w3.org/TR/xmldsig-core`.

Xhafa, F., Barolli, L., Daradoumis, T., Fernàndez, R. and Caballé, S. (2009). JXTA-Overlay: An interface for efficient peer selection in P2P JXTA-based systems, *Computer Standards and Interfaces*, vol. 31, no. 5, pp. 886 – 893.