A security layer for JXTA core protocols

Joan Arnedo-Moreno[1]and Jordi Herrera-Joancomartí[2]
(1) Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya, Rb. Poble nou 156, 08018 Barcelona
jarnedo@uoc.edu
(2) , Escola Tècnica Superior d'Enginyeria, Universitat Autònoma de Barcelona, Campus de Bellaterra
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 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.

I. Introduction

In a peer-to-peer 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 [1] that all peers have equivalent capabilities, and a central server with more processing power is no longer necessary. JXTA [2] 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 providing the necessary services in order for any other protocol to be used in JXTA applications and operate within the network. Peers may use such protocols in order to advertise and discover resources, join peer groups and dynamically route messages across multiple network hops.

Since, ultimately, every JXTA application must rely on the use of its core protocols in order to interact with other peers within the network, it is very important to provide capable methods to secure them in order to avoid possible attacks. The current JXTA reference implementation addresses some of this 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 relies on the existence of a party that must be trusted by all peer group members.

The standard security threats to be addressed under a P2P environment can be divided into two different groups: passive and active ones [3]. Passive attacks are those in which the attacker just monitors activity and maintains an inert state whereas in active attacks, communications are disrupted by the deletion, modification or insertion of data.

The contribution of this paper is a modular method to protect JXTA's core protocols against both type of attacks in a manner specifically suited to their 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 the peer-to-peer model pure, and authenticity is locally decidable (eliminating the possibility of collusion). This is capital in an environment where messaging may be continuous and other peer's availability is not guaranteed.

This paper is organized as follows. Section II provides an overview of JXTA core protocols and the current methods for securing them. Section III describes the proposal for improving the current methods, by following the JXTA ideary of XML message formatting. Concluding the paper, section IV summarizes the paper contributions and outlines further work.

II. An overview of JXTA core protocol security

JXTA's endpoint communication is structured in a classical layered approach, core protocols acting as a gateway to networking operations under a peer-to-peer environment (see Figure 1). 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 wire transport protocols at the Wire Transport layer, such as TCP, HTTP, TLS or multicast. The message created at the Messaging layer is considered the application level data to be sent by the wire transport protocol.

A message is essentially a set of name/value pairs, organized as an ordered sequence, the most recently added element appearing 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. All core protocols are codify messages as XML data, the message name being the root XML element tag and its value the corresponding XML subtree.

It is not mandatory for JXTA implementations to deploy all core services, but at least PDP and ERP must be supported in

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.

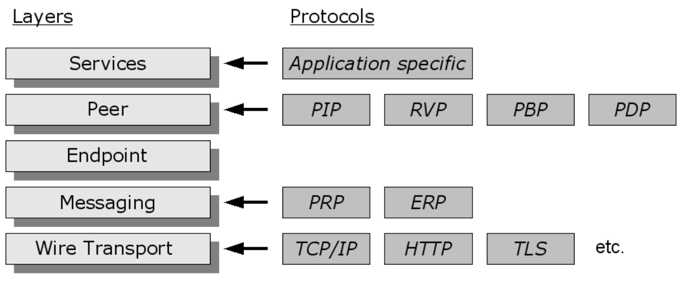

Fig. 1. JXTA protocol layers and protocols

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 [4] supports all six of them.

In the current reference implementation, messaging has been secured assumping that the Personal Security Environment (PSE) acts as the group's Membership Service. The Membership Service is one of JXTA's core services, providing group membership and identity management within a peer group by providing each group member with a credential, which may be used to provide proof of group membership.

The PSE's credentials are based on PKIX [5] 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 [6] protocol, and at the wire transport layer, via its own definition of TLS [7].

A. Messaging layer security

The 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 ultimately defines a wire transport protocol in JXTA. CBJX pre-processes messages in order to generate a secure encapsulation, resulting in 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 definition shown in Listing 1, 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.

XML Listing 1 - 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>
```

In order to generate both signatures, XML data is serialized and processed as plain text by the signature algorithm. An overview of message encapsulation is shown in Figure 2. CBJX encapsulates signatures by using a single *Signature* element containing a Base64-encoded PKCS#7 [8] binary signature. Once the CBJX message is complete, it is sent just like a standard message, via the wire transport layer.

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

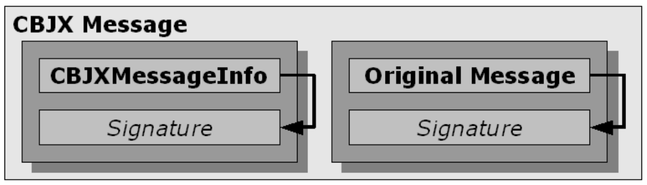

Fig. 2. CBJX secure encapsulation

Apart from digital signatures, CBJX provides an additional lightweight authenticity method by using Crypto-Based Identifiers (CBIDs [9]). This method provides authentic messaging without the need of certificates issued by a TTP (Trusted Third Party). Self-signed certificates are good enough.

B. 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.

TLS provides connection security with two basic properties: privacy and integrity. Privacy is achieved by using symmetric cryptography for data encryption (e.g., DES, RC4, etc.) The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated using a handshake protocol. Integrity is obtained by means of a message integrity check using a keyed MAC. Secure hash functions are used for MAC computations. These properties provide security against both active and passive attacks.

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.

III. An interoperable modular security approach

From the explanation in section II, 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 application interoperability has to be maintained. Signatures are generated by serializing XML as plain text, which is not ideal for XML processing, as will be explained in section III-B. Furthermore, under the current secure protocols, 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 messages.
- No privacy at the messaging layer: Currently, the only way to achieve data privacy 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 transparently secure neither other current JXTA wire transport protocols such as message propagation via multicast or HTTP proxying, nor any future ones (for example, UDP or RTP).
- TTP-based trust model: The use of 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 to use X509 certificates and a TTP based trust model. This is not always desirable in a dynamic and decentralized environment such as peer-to-peer, specially when trying to maximize peer equality and self-organization. Furthermore, the use of a TTP inherits additional problems which increase system's complexity like, for instance, certificate chain management and revocation.

In order to solve this issues, we move a step further from the security proposal in [10] in order to deploy a security layer at the Messaging layer, which is common to all of JXTA's core protocols. We define a data encryption and signature format for message elements based on XML standards, making use of the standard JXTA messaging signature capabilities at the Messaging layer as explained in subsection II-A. Using this method, the message format, as defined in the JXTA v2.0 protocols specification [11], is maintained.

This approach allows peers which do not support signature to process messaging in a transparent way. As a result, each peer may choose its own degree of security without being constrained by other peer's decision on that regard. Due to JXTA's layered protocol architecture, deploying security at the Messaging layer ensures that this proposal may easily integrate with existing applications, since this layer is completely transparent to application data. This approach also allows different secure protocols to coexist, both current ones (CBJX and TLS) as well as future proposals, so applications may choose the one that suits its specific needs.

We also take advantage of the same CBID format and secure key distribution as proposed in [10] to guarantee public key authenticity. It must be noted that advertisement security 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.

A. 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 [12], [13], [14]. For our proposal, we specifically use the *xmlenc* [14], since in conjunction with its brother standard *xmldisg* [15], both provide a full security set for passive and active attacks against XML data. Since all protocols in JXTA are XML-formatted, it is a logical election. 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 [12] and providing a reasonable result document size, in contrast with [13].

Messages are selectively encrypted using a wrapped key encryption scheme (such as the one defined in [16]). 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, that can only be accessed by only one of the recipients.

This scheme is applied at the message layer 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 represented by 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.

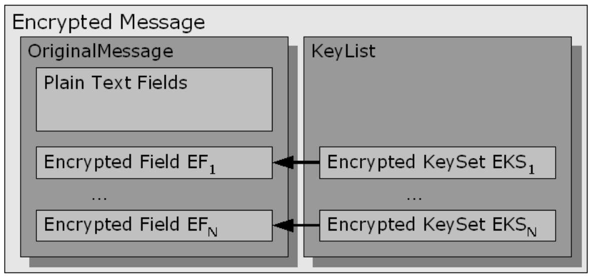

Fig. 3. Message encryption profile

Each wrapped key within a keyset is defined by an xmlenc *EncryptedKey* element and contains all 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. Peers 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.

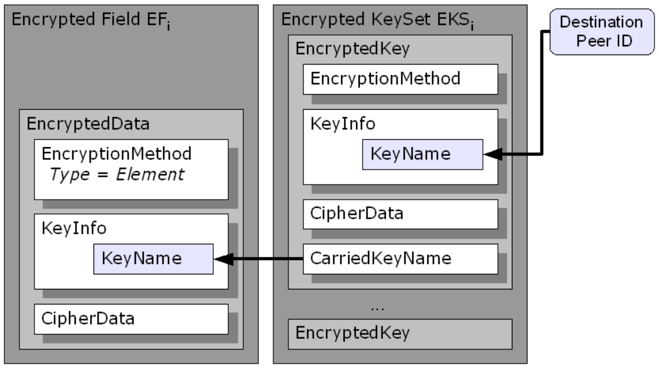

Fig. 4. Xmlenc encryption profile

By using this XML profile and message schematics, it is possible to accommodate selective entry encryption. Since the final encrypted data is sent to wire transport protocols as a standard JXTA message, encrypted fields become transparent to its basic operation.

1) Encryption and decryption process: The process of message encryption that generates the xmlenc profile previously defined can be described as follows.

Encryption:

1) Peer A needs to send a core protocol message.
2) A new *KeyList* element is generated.
3) 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.
4) 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. Its *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.
5) 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) exists within the *KeyList* element which may decrypt it.
6) The message is sent via the chosen wire transport protocol.

A sample encrypted message after this process is shown in Listing 2 (some ID's and Base64 encoded data have been shortened in order to improve readability). Specifically, this message contains a PRP query. 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 demonstrates.

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 *EncryptedKey* elements, EK, which contain B's Peer ID in its *KeyInfo* field.
4) For each *EncryptedKey*, Enc_i, in EK:
    a) The encrypted field of the message, EF_i, to be processed 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 the *CipherData* field of Enc_i .

XML Listing 2 - 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>Message Key ID</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>Message Key ID</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>Message Key ID</CarriedKeyName>
      </xenc:EncryptedKey>
    </KeySet>
  </KeyList>
</xmlsecure:EncryptedMessage>
```

c) The original EF_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 he could satisfactorily decrypt.

B. Core protocol authenticity and integrity

Authenticity and data integrity (as well as non-repudiation) are provided to core protocols by using XML signature (xmldsig) at the messaging layer. Since the process is very similar to the method proposed in [10] for advertisement authenticity, a general outline will be presented. However, it has been specifically adapted for core protocol messaging.

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 [17]. 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). Only for that reason, xmldsig is capital 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, and 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 elements (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 and ignore the signature).

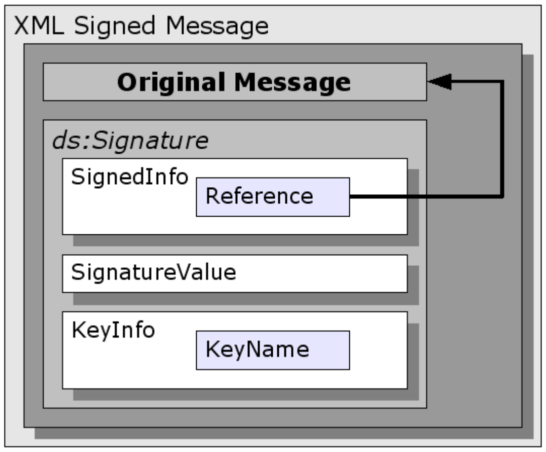

Fig. 5. Xmlsig detached signature profile

As a detached signature, in this scenario it is enough to use a default URI in the *Reference* element in order 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.

1) Signature and validation process: 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 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.

The reader can see [10] for the details about how the key is retrieved and the CBID is generated in steps 1 and 2.

A sample message signature is shown in Listing 3 (some ID's and Base64 encoded data have been shortened). Notice that, since it is a detached signature, it is not necessary for the message to be present. The fact that it is a detached signature can be noticed since an implicit URI is used at the *Reference* element.

XML Listing 3 - 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>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>
```

Both xmlenc and xmldsig validation nicely integrate because 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 an xmlenc *Transform* element within the signature *Reference* element.

IV. 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, 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 peer-to-peer, 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 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.

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, it is possible communication between peers which choose to apply security and those who 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 peer-to-peer environment.

At this stage of research, further work goes toward experimentally evaluating the approach, focusing in the study of its impact on peer performance compared to current approaches (both CBJX and TLS).

References

[1] Andrew Oram, *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001.
[2] Sun Microsystems, "Project JXTA", 2001, `http://www.jxta.org`.
[3] Brookshier D. Govoni D., Soto J.C. and Krishnan N., "Jxta and security", *JXTA: Java P2P Programming*, pp. 251–282, 2002.
[4] "Jxta 2.5 rc1", June 2007, `http://download.java.net/jxta/build`.
[5] CCITT, "The directory authentication framework. recommendation", 1988.
[6] D. Bailly, "Cbjx: Crypto-based jxta (an internship report)", pp. 108–109, July 2002.
[7] T. Dierks and C. Allen, "Ietf rfc 2246: The tls protocol version 1.0", 1999, `http://www.ietf.org/rfc/rfc2246.txt`.
[8] Kaliski B., "Pkcs#7: Cryptographic message syntax version 1.5", 1998, `ttp://www.ietf.org/rfc/rfc2315.txt`.
[9] Montenegro G. and Castelluccia C., "Crypto-based identifiers (cbids): Concepts and applications", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, 2004.
[10] Joan Arnedo-Moreno and Jordi Herrera-Joancomartí, "Persistent interoperable security for jxta", in *Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC) 2008*. 2008, pp. 354–359, IEEEPress.
[11] Sun Microsystems Inc., "Jxta v2.0 protocols specification", 2007, `https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html`.
[12] Kudo M. Hada S., "Xml access control language: Provisional authorization for xml documents", 2002, `http://www.research.ibm.com/trl/projects/xml/xss4j/docs/xacl-spec.html`.
[13] Geuer-Pollmann C., "Xml pool encryption", *XMLSEC '02: Proceedings of the 2002 ACM workshop on XML security*, 2002.
[14] W3C, "Xml encryption syntax and processing", 2002, `http://www.w3.org/TR/xmlenc-core/`.
[15] W3C, "Xml-signature syntax and processing", 2002.
[16] J. Staddon B. Kaliski, "Pkcs1: Rsa cryptography specifications. version 2.0", 1998.
[17] J. Boyer, "Canonical xml. version 1.0", 2001, `http://www.w3.org/TR/xml-c14n`.