Persistent interoperable security for JXTA advertisements

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

***Abstract*— In order to access resources within a peer-to-peer network, JXTA completely relies in the usage of advertisements published by the resource owner. This may easily lead to DoS or spoofing attacks from malicious peers by forging advertisements with false identifiers unless advertisement authenticity is provided. Furthermore, in a fully distributed environment, in order to truly isolate a malicious node, collaboration is necessary from other peers. For that reason, it must be possible to prove to other parties that a peer is being disruptive. This paper presents a method that provides authenticity and non-repudiation to JXTA advertisements. This scheme is fully distributed and based in 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, distributed systems, messaging, Java, xmldsig.

I. Introduction

A peer-to-peer environment is a virtual network where all involved parties share resources and collaborate in order to provide basic services, such as content, processing or messaging. 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 protocols that enable the creation of peer-to-peer networks.

JXTA differs from other peer-to-peer protocols (such as Gnutella [3]) because introduces the concept of *peer group*, one of the main foundations of its architecture. Usually, peer-to-peer environments are conceptualized as a global overlay network without any kind of logical segmentation or segregation as far as resource availability is concerned. However, in JXTA the global overlay network is segmented into hierarchical groups of peers, which offer a context for accessing services. Peers may access a specific resource only if they previously join the group that offers such service. The concept of peer groups is very important, since all JXTA security services are ultimately related to peer group membership management.

Resources must first be somehow publicized to all group members before peers become aware of their availability. In JXTA, resources are announced using a special message named *advertisement*, which is sent to all other peer group members. Being the main gateway to all resources within a peer group, it is very important to secure advertisements to avoid possible attacks.

One of the security threats in this environment is the possibility of spoofing peer identifiers. Peers may publish bogus resources with random identifiers (or even worse, some other peer's identifier), creating a denial-of-service attack on the network by filling it with rubbish. It is not possible to avoid the publication itself, since the communication channel is always open, but it should be possible to quickly recognize which peers are trying to actively disrupt the network, in order to isolate them or expel them from the peer group.

The current JXTA reference implementation addresses this problems by securing messaging. However, the provided method is not fully satisfactory, as it does not fully comply with the JXTA specifications ideary of XML data formatting and relies on the existence of a party that must be trusted by all peer group members.

The contribution of this paper is a lightweight method for providing authenticity and non-repudiation specifically suited to the idiosyncracies of JXTA advertisements. Authentic advertisements guarantee that it is not possible to spoof some other peer's identifier in order to avoid responsibilities. The presented method does not rely on external parties, keeping the peer-to-peer model pure. Authenticity is locally decidable (eliminating the possibility of collusion). This is very important in an environment where advertisement traffic is continuous and other peer's availability is not guaranteed. Furthermore, it is not mandatory to validate advertisements at the time of reception, making possible deferring validation until the advertisement must be really used. Non-repudiation mechanisms for advertisements are also important since they allow proving to a third party (another peer within the peer group) that a peer published a false advertisement. In a pure peer-to-peer environment, non-repudiation information can be used collaboratively by peers in order to isolate malicious nodes that publish false advertisements.

This paper is organized as follows. Section II provides an overview of JXTA advertisements 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

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

IV summarizes the paper contributions and further work.

II. An overview of JXTA Advertisements

Advertisements are meta-data records used by JXTA protocols to describe all available resources in the network, from peers to group services. Peers cannot access a resource without previously retrieving its associated advertisement. The most important types of advertisements in JXTA are the following ones:

- Peer Advertisement: Describes a peer and the resources it provides to a peer group.
- Peer Group Advertisement: Describes a peer group, its specific resources and its offered services' parameters.
- Module Class Advertisement: Provides a description of what a particular Module Class ID stands for. A Module Class ID is what some code running on JXTA uses to designate modules which it depends upon.
- Rendezvous Advertisement: Describes a peer that acts as a Rendezvous Peer for a given peer group. Rendezvous are super-peers, which create a logical super-network within the JXTA network itself, used in order to route and index messages.
- Pipe Advertisement: Describes a pipe, the JXTA core mechanism for exchanging messages between two applications or services, providing a simple, unidirectional and asynchronous communication channel.

All advertisements are codified using XML and passed between peers using the JXTA protocols. 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 Peer Advertisement is shown in Listing 1 as a sample of advertisement format.

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>

<xs:complexType name="serviceParam">
  <xs:sequence>
    <xs:element name="MCID" type="jxta:JXTAID"/>
    <xs:element name="Parm" type="xs:anyType"/>
  </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>
```

Advertisements can be published using two different methods: local and remote publication.

In *local publication*, the advertisement is indexed and stored in the peers local cache. Following, the advertisement's index is pushed to the rendezvous peer for that group and is then distributed and replicated between all rendezvous in the global super-network peers using the Shared-Resource Distributed Index (SRDI) service [4], [5]. The rendezvous network acts as a remote index cache.

Using this method, it is possible for peers outside the local network (out of broadcast range) to retrieve group advertisements by asking its rendezvous peer. It also enables group members which where off-line for some time to retrieve advertisements published during its disconnection. Whenever a peer receives the advertisement, it is indexed, stored in a local cache and assigned an expiration date.

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 (the resource is not ultimately tied to the publishing peer's availability). 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.

A. Current security in JXTA advertisements

The Advertisements' description presented in the previous section shows that they are one of the building blocks of the JXTA architecture. In the case that false advertisements are propagated and stored through the network, it will stop functioning properly. In the JXTA base specification, where no security is applied, it is trivial for a malicious or malfunctioning peer to spoof false advertisements with bogus ID's or information. Even worse, it is possible to publish advertisements specifying that another existing peer has some inexistant service.

However, in the current reference implementation, messaging has been secured when the Membership Service is implemented using the PSE (Personal Security Environment). 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 [6], provides three basic Membership Services. However, the JXTA specification lets the door open for defining new types of custom Membership Services.

The PSE Membership Service is the default one in JXTA. Credentials are based on PKIX [7] 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 advertisements may be secured at two different (but not disjunct) levels: by using transport level security, or by securing the advertisement at application level.

1) Transport level security: The JXTA specification guarantees end-to-end transport security via two different protocols: TLS (Transport Layer Security [8]) and CBJX (Crypto-Based JXTA Transfer [9]). Both protocols are not specifically tied to advertisement security and may be applied to any message exchange, providing different degrees of security: TLS provides private, mutually authenticated, reliable streaming communications, whereas CBJX provides lightweight secure message source verification. This section will focus on CBJX, since it is a JXTA-specific protocol.

CBJX uses digital signature in order to provide integrity and authentication. It adds an additional information block to the secured message, as shown in Listing 2: a `PeerCert` element, which contains the source peer certificate, both the source and destination addresses, and the source peer ID (the JXTAID type definition was already shown in listing 1). Both the message body and the cryptographic information block are digitally signed, generating two separate signatures. The certificate inside the cryptographic information block is used to validate both signatures.

In order to generate both signatures, XML data is serialized, processed as plain text, and fed to the signature algorithm.

XML Listing 2 - CBJX crypto-information XML schema

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

Apart from digital signature, CBJX provides lightweight authenticity by using Crypto-Based Identifiers (CBIDs [10]), which will be described in more detail in section III. At this point, only notice that this method provides authentic messaging without the need of certificates issued by a TTP (trusted third party). In contrast, TLS needs TTP issued certificates in order to function.

However, in both cases (TLS and CBJX), information security is only provided during transit by protecting the JXTA transport protocol at a lower layer. Once the transport encapsulation is removed and information is stored into the local peer cache, it is no longer secured. It also must be taken into account that both types of transport methods do not support full advertisement propagation (used in remote publication, as explained in section II-A), they only support end-to-end communications.

2) Advertisement level security: As a means to achieve end-to-end security, the reference implementation adds applies an optional security layer to advertisements by specifically signing them. No distinction between different types of advertisements is made, all use exactly the same format when signed: the Signed Advertisement.

The XML schema definition for a Signed Advertisement is shown in 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 [11]. 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.

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>

<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>
```

Since signed advertisements are uniquely based on certificates under the PSE Membership Service, authenticity can only be provided by using a TTP that will issue such certificates, sharing the same constraints as TLS transport security.

III. Providing persistent interoperable advertisement security

From the explanation in section II-A, the current security layer in advertisement publication has two main shortcomings which could be improved:

- Lose of interoperability: The format of signed advertisements is completely different from the format of the original advertisement, which means that an application that does not support signed advertisements will not be able to recognize them as such. The original advertisement also

loses its XML structure and becomes Base64 encoded plain text during transit. Even though JXTA makes heavy use of XML in its protocols, no XML standard is used for signature generation, which is an additional hurdle if interoperability is to be maintained. This shortcoming also applies to JXTA transport protocols, which generate signatures by serializing XML as plain text.

- Transient security: Advertisements are only secured during transport. Since they are stored into the local cache until their expiration, they should also be kept secure there. The transient nature of the security layer also forces the application to verify all messages upon immediate reception, before encapsulation is discarded, even though most of them might never be used (its associated resource never accessed), impacting performance.

Additionally, some of the current security services need a TTP in order to properly issue the digital certificates that ensure advertisement authenticity. The use of a TTP goes against the spirit a pure peer-to-peer model, where it is assumed that peers are equal and self-organizing, not utterly reliant on other specific peers. Furthermore, the use of a TTP inherits additional problems which increase the system's complexity, such as that of certificate chain management and revocation.

All those issues are solved in our proposal. On one hand, we define an extension format for Signed Advertisements based on XML signatures, maintaining the base structure of advertisements as defined in the JXTA v2.0 protocols specification [12]. In this way, peers which do not support signature may nevertheless process messaging in a transparent way. Peers may freely choose which advertisements should be secured, and chose its own degree of security, without impacting other peers. On the other hand, the use of a TTP is avoided by using Crypto-Based Identifiers (CBIDs).

The concept of CBIDs, or statistically unique and cryptographically verifiable IDs (SUCV IDs), was initially conceived for IPv6 addressing in order to solve the issue of address ownership, avoid router supplantation attacks and binding update packet spoofing [13], [14]. Using this mechanism, each address is automatically bound to a specific node.

Under a CBID scenario, each node generates a private/public key pair. Since each node is identified by its address, some method that binds the key pair to the address is necessary in order to provide authenticity. In the absence of a TTP, this binding is created via applying a pseudo-random function on the public key. The result (or part of it) is henceforth used as the node address. Any message sent by a peer is then signed using its private key.

In order to validate CBID ownership, the messages's signature is validated. If validation is correct, its is proved that the source peer holds the associated private key. Then the validating public key is used to generate the source address, *i.e.* the CBID. If the obtained address is the same as the claimed source address, the message is authentic. This method can be easily ported to JXTA IDs instead of IPv6 addresses. By using this method, ID ownership, and therefore authenticity, is guaranteed in a secure manner, providing countermeasures against DoS and ID hijacking [15].

Other methods exist in order to bind key pairs to identifiers without the need of a TTP, such as the id-based [16] approach or self-certifying keys [17]. However, since JXTA IDs must follow a very specific format, they are not feasible.

A. Peer CBID Generation

Some guidelines must be taken into account when creating Peer ID's. According to the JXTA Protocols Specification, a Peer ID should canonically, uniquely and unambiguously refer to a peer. Furthermore, it should be possible to determine the associated Peer Group ID, identifying the peer group of which the peer is a member, from a Peer ID. Furthermore, it must follow the format defined by the specification.

A Peer ID is 64 bytes long and is always formed by the following fields: Peer Group UUID [18] (16 bytes long), Peer UUID (16 bytes long), and the rest of bytes set at zero up to the 64th byte, which specifies the advertisement type (its value being set at 03 for Peer ID's). Notice that only 128 bits, the Peer UUID, are really unique to each peer within the same peer group. For that reason, those are the only values which can be manipulated in order to generate CBID's.

A simple way to achieve a CBID from the public key is by applying the SHA-1 [19] hash algorithm on it. The result is considered the peer UUID in order to construct the full JXTA ID. Since SHA-1 produces 160 bits and only 128 are needed, the first 32 most significant bits are ignored. This method is the one applied by the CBJX transport protocol when generating CBIDs.

Using this method of CBID generation, no certificates are needed, the whole system may be based on raw private-public key pairs. In the case that certificates must be used for some reason, self-signed certificates are enough. An additional advantage of this method for generating CBIDs is that since it is based in the SHA-1 hash algorithm, it applies under the UUID type 5 specification [18]. This method also allows to integrate peers which use CBIDs with those who don't, since their IDs are random numbers anyway.

It must be taken into account that a single ID is generated from a specific public key, which might be problematic if two different publics keys produce the same UUID field. However, for a population of $1.2 * 2^n/2$ peers, the probability of UUID collision is about 50%[20]. For $n = 128$ the resulting value is statistically more than acceptable. Furthermore, because of the JXTA ID format, it would be necessary that both peers are members of the same group in order to produce a Peer ID collision.

Binding peer IDs with public keys implies that changing ID means changing the public key (and viceversa). This property may be seen as an advantage since it provides an incentive to not continuously change identity, because under certificate-base group membership (such as PSE), a peer changing its public key (or its ID) stops being a member of all groups and must start from scratch. In reputation- based systems (such as Poblano [21]), all reputation is lost.

B. Peer ID binding publication

Once a peer has generated a proper ID, its public key has to be distributed to other peer group members in order for

them to properly validate its CBID. This is achieved using the Peer Advertisement, by adding some additional fields. In this way, no extra message types or additional protocols are necessary, seamlessly integrating with the current capabilities of JXTA. Those peers which do not support advertisement security will ignore additional fields, but will still recognize Peer Advertisements as such. Using the Peer Advertisement also provides an autonomous manner in order to manage ID binding publication.

The XML schema for the new format of Peer Advertisement that enables the publication of key binding to a specific Peer ID is shown in Listing 4. The JXTAID defined in the `PID` element must be generated according to the guidelines detailed in the previous subsection. Type definitions from the *jxta* namespace are defined Listing 1. The *ds* namespace is very extensive and its data type definitions will not be copied in this paper. They are thoroughly specified in [22].

XML Listing 4 - Key binding via 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:element name="Signature" type="ds:SignatureType"/>
    <xs:element>
  <xs:sequence>
</xs:complexType>
```

XML signature (xmldsig) [22] is used as a means to sign Peer Advertisements, by using an encapsulated signature type. Peer Advertisements will always contain an XML signature with a `KeyInfo` element which explicitly encapsulates the public key (such as a `KeyValue` or `X509Certificate` elements). Nevertheless, it is important to remark that this advertisement just makes public such binding, but the binding itself is ultimately achieved via the use of CBIDs. Whenever a Peer Advertisement is received, the binding validity may be tested in order to ensure its correctness.

Using xmldsig is a logical approach, since it is a standard for signing XML and all protocols in JXTA are XML-formatted. 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 [23]. 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 does not take this fact into consideration, since a single different bit, however irrelevant to the XML semantics, will invalidate a signature. Just for that only reason, xmldsig is better 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. That means that it is easy to integrate with the current PSE group membership credentials just by defining a new URI type that can be used as a xmldsig KeyInfo type (such as, for example, *http://jxta.org/jxta#PSE*). Any application which recognizes such URI may henceforth directly use the `PSECred` element (as defined in listing 3) in a xmldsig `KeyInfo` element.

Using a CBID directly in Peer Advertisements also allows benefiting from all its advantages under message propagation (which CBJX does not support) when forcing remote publication of advertisements. In such scenario, it is also possible to maintain a proactive stance against ID spoofing by filtering bad messages during transit(for example, in rendezvous peers), since just validating a CBID is computationally easy (comparing a single hash result). This minimizes garbage traffic and easily pinpoints attackers.

C. Advertisement authenticity and non-repudiation

Authenticity and non-repudiation (as well as integrity) is provided by using xmldisg in all advertisements.

Once a peer receives a signed advertisement, it can be stored into the peer's cache and its verification can be deferred up to the moment the advertisement should be used. When the advertisement has to be validated, the following steps must be performed:

1) Retrieve the source peer public key.
2) Apply the SHA-1 hash algorithm to the public key. Generate a JXTA CBID from the result as described in subsection III-A.
3) Compare the resulting CBID to the source peer CBID. If equal, advertisement authenticity is proved.
4) Validate XML signature using the public key retrieved in Step 1. If valid, integrity and non-repudiation are proved.

In Step 1, the source peer public key must be retrieved. Notice that the public key is not necessarily included in each signed advertisement, it is enough to be included in the Peer Advertisements. In order to retrieve the Peer Advertisement (and the corresponding public key), the source peer ID can be obtained from the xmldsig `KeyInfo` element of the signed advertisement, which can then be used to find the Peer Advertisement in the local cache. In the case that a peer did not previously receive the sender's Peer Advertisement and it cannot be found in the local cache, it may be easily retrieved by asking the sender or a Rendezvous Peer via the JXTA Discovery Protocol (specifically, by using a type 0 query). Using this protocol, any peer may retrieve any advertisement within the group.

Keeping the peer's public key in the Peer Advertisement is not inconvenient for advertisement signature validation since, under the assumption of deferred validation, an advertisement will be validated when the resource it represents must be accessed. In the case that such Peer Advertisement is unreachable, that would mean that the source peer is offline and the resource to be accessed will be unavailable anyway. Nevertheless, by forcing remote publications, the Peer Advertisement may be explicitly pushed to group members.

A sample signed Peer Group Advertisement is shown in Listing 5 (some ID's and Base64 encoded data have been shortened).

XML Listing 5 - Signed Peer Group 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 Advertissement</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>
```

IV. Conclusions and further work

A new proposal for advertisement security in JXTA been presented. Its main contributions are twofold.

First of all, the proposed method provides advertisement authenticity and non-repudiation in a lightweight manner, without the need of a TTP. Furthermore, it maintains end-to-end advertisement security for its whole lifetime, not just during transport, keeping its security layer even when stored in a peer's locally cache, until expiration. Even though JXTA may use its current implementation of TLS or CBJX in order to provide a security layer to messaging, such security is transient. This is achieved by upgrading CBIDs from the transport layer to the advertisement itself. Using this approach, it is also possible to use CBIDs in advertisement propagation, which is not currently possible.

Finally, our proposal keeps interoperability by maintaining the advertisements base format, instead of creating a completely new one, and provides a method to efficiently publish CBID-key binding by taking advantage of JXTAs own capabilities, without the need of additional protocols. Our approach makes it possible to seamlessly integrate peers which support advertisement security which those who don't (or chose not to).

An additional benefit of maintaining the advertisements base format is that it is no longer necessary to pre-process data at the reception time before being stored into the cache. It is possible to defer validation until the moment the advertisement has to be used. This feature, added to the use of CBIDs, guarantees that the protocol will be lightweight, since only advertisements that will be necessary must be validated, and its validation need not be at the moment of reception. This feature may improve the peer's performance since there may be a lot of advertisement traffic in a given network that the peer will not use and that it does not need to be validated.

Further work goes toward using the provided non-repudiation mechanisms to define an incrimination protocol to inform to group members that a specific peer is publishing false advertisements. Using collaboration mechanisms, such peer should be isolated and expelled from the group, its traffic being ignored.

References

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

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

[3] "Gnutella", http://rfc-gnutella.sourceforge.net.

[4] M.Abdelaziz M.Duigou C.Haywood J.C.Hugly E.Pouyoul B.Yeager B.Traversat, A.Arora, "Project jxta 2.0 super-peer virtual network", Tech. Rep., SunMicrosystems,Inc, May 2003.

[5] Abdelaziz M. Traversat, B. and E. Pouyou, "Project jxta: A loosely-consistent dht rendezvous walker", Tech. Rep., SunMicrosystems,Inc, March 2003.

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

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

[8] Allen C. Dierks, T., "Ietf rfc 2246: The tls protocol version 1.0", 1999, http://www.ietf.org/rfc/rfc2246.txt.

[9] D. Bailly, "Cbjx: Crypto-based jxta (an internship report)", pp. 108–109, July 2002.

[10] Castelluccia C. Montenegro, G., "Crypto-based identifiers (cbids): Concepts and applications", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, 2004.

[11] Ed. S. Josefsson, "Ietf rfc 3548 - the base16, base32, and base64 data encodings", 2003, http://www.ietf.org/rfc/rfc3548.txt.

[12] Sun Microsystems Inc., "Jxta v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[13] T. Aura, "Cryptographically generated adresses (cga)", http://www.ietf.org/rfc/rfc3972.txt.

[14] Luciano Bononi and Carlo Tacconi, "Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks", *Int. J. Inf. Secur.*, vol. 6, no. 6, pp. 379–392, 2007.

[15] P. Nikander, "An address ownership problem in ipv6", 2001, draft-nikander-ipng-address-ownership-00.txt.

[16] A. Shamir, "Identity-based cryptosystems and signature schemes.", *Advances in Cryptology - CRYPTO '84*, pp. 47–53, 1984.

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

[18] Mealling M. Salz R. Leach, P., "A universally unique identifier (uuid) urn namespace", http://www.ietf.org/rfc/rfc4122.txt.

[19] NIST, "Fips pub 180-1: Secure hash standard", 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[20] Van Oorschot P. Vanstone P. Menezes, A., *Handbook of applied cryptography*, CRC Press, 1997.

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

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

[23] J. Boyer, "Canonical xml. version 1.0", 2001, http://www.w3.org/TR/xml-c14n.