JXTA resource access control by means of advertisement encryption

Abstract

JXTA is an open-source initiative that provides a middleware for the creation and deployment of peer-to-peer (P2P) applications. Resources in a JXTA network are accessed through *advertisements*, special metadata documents published by its owner. By controlling access to advertisements, it is also possible to restrict access to resources. However, in an overlay multihop network, the final recipient of an advertisement is not the only one that obtains it, since intermediate peers may copy it. This paper presents a method for JXTA advertisement encryption in order to effectively control access to published resources. The scheme is fully distributed, based on a P2P pure model and does not disrupt core discovery services. The advertisement publisher controls which information should be globally disclosed and which one will be restricted to only a particular subset of peers.

Key words: peer-to-peer, security, peer group, JXTA, distributed systems, XMLenc, privacy.

1 Introduction

A peer-to-peer (P2P) 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 that all peers have equivalent capabilities [1], and a central server with more processing power is no longer necessary. JXTA [28] is a set of open protocols that enable the creation of collaborative P2P applications.

JXTA differs from other P2P protocols (such as Gnutella [12]) because the concept of *peer group* is introduced, one of the main characteristics of its architecture. Usually, P2P 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 peer groups, 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 published to all group members before peers become aware of their availability. In JXTA, resources are announced to other peer group members using special messages named *advertisements*. Any resource within a peer group can be located and used once a peer obtains the corresponding advertisement.

In this type of collaborative environments, security threats in resource publication and discovery are a concern [25] since the possibility exists that an unauthorized peer obtains a particular advertisement, and thus can access the corresponding resource. This is an important concern since, as an overlay multihop network, JXTA advertisements are freely transmitted across unknown peers and may be stored in peers which are not its original creator (for example, in order to provide redundancy). In this scenario, it is not possible to effectively control advertisement dissemination without providing some degree of data privacy. Even though it is not possible to control which peers receive the advertisement, it is feasible to control which peers will be able to effectively use its content. However, a compromise between data privacy and accessibility must be reached, since some advertisement information should not be disclosed whereas some other advertisement information should be freely accessible in order to allow core discovery services to locate advertisements within the network.

The current JXTA reference implementation deals with security issues in general messaging by providing methods for data integrity, authenticity and non-repudiation, but is does not offer any mechanism to control access to advertisement content. Even though the current JXTA implementation provides private secure messaging at transport layer, any peer which is the intended receiver in advertisement publication has access to its full content. In addition, the provided method for secure messaging is not completely satisfactory, as it does not fully comply with the JXTA specifications of XML data formatting.

The main contribution of this paper is a method for providing privacy to advertisements in order to control access to peer group resources. The proposed method is specifically suited to the idiosyncrasies of JXTA advertisements by keeping the original XML format and interoperating with core discovery services, so it is still possible to locate resources and their owner in a transparent way. The presented method does not rely on external parties, keeping the P2P model pure. This is very important in an environment where advertisement traffic is continuous and other peer's availability is not guaranteed. Furthermore, the proposal does not force total advertisement decryption at the reception time, making it possible to defer processing until the advertisement must be really used, as well as securing data at a local storage level.

Finally, our proposal nicely integrates with the security mechanisms described in [14], providing a joint solution in order to deploy basic security services on advertisements: privacy, integrity, authentication and non-repudiation.

This paper is organized as follows. Section 2 provides an overview of JXTA advertisements and the current methods for securing them. Section 3 describes our proposal to provide privacy to parts of the advertisement content, keeping the JXTA XML message format. Concluding the paper, Section 4 summarizes our paper contributions and outlines further work.

2 An overview of JXTA Advertisements

Advertisements are meta-data records used by JXTA protocols to describe any available resource 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. Each peer is responsible for its own Peer Advertisement publication.
- *Peer Group Advertisement*: Describes a peer group, its specific resources and its offered services' parameters.
- *Module Class Advertisement*: Provides the description of a Module Class ID. A Module Class ID is used by any service code running on JXTA to designate the modules it depends on.
- *Rendezvous Advertisement*: Describes a peer that acts as a *rendezvous peer* for a given peer group. Rendezvous are super-peers that route and index messages.
- *Pipe Advertisement*: Describes a *pipe*, the JXTA core mechanism for exchanging messages between two applications or services. Pipes provide a simple, unidirectional and asynchronous communication channel.

All advertisement types are codified using XML and passed between peers using the JXTA core protocols. Using XML codification allows programming language/platform independence, it offers a self-described specification and it is able to enforce correct syntax. As an additional feature, XML can easily be translated into other encodings, such as HTML, which may allow to access resources to peers that do not support XML. The XML schema for a Peer Group Advertisement is shown in Listing 1 as a sample of advertisement format.

In this sample, both the `GID` and `MSID` fields are mandatory, indicating the peer group and specification identifiers respectively. The rest of fields are optional, providing a human-readable name and a description of the peer group. The `Svc` fields are of special interest for this work since they provide a list of the

services offered by the peer group and the required initialization parameters in order to access them.

XML Listing 1 - Peer Group Advertisement

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

<xs:complexType name="PGA">
  <xs:sequence>
    <xs:element name="GID" type="jxta:JXTAID"/>
    <xs:element name="MSID" 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>
```

2.1 Advertisement publication

Advertisements can be published using two different methods: local publication and remote publication. Depending on which method is used, it may be possible for other peers, different than the original creator or the intended destinations, to access the published resource.

In *local publication*, the advertisement is indexed and stored in the peer's local cache with an assigned expiration date. Following, the advertisement's index is pushed to the rendezvous peer for that group and then distributed and replicated through all rendezvous peers in the global peer super-network using the Shared-Resource Distributed Index (SRDI) service [5,6]. 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. The SRDI also enables to retrieve advertisements to group members which were off-line during the time that the advertisement was published.

Even though during the SRDI publication process the original advertisement is distributed to rendezvous peers (which means that such peers have access

to the advertisement's full content at some point), only a copy of its index is eventually stored. Rendezvous peers only provide a mechanism to identify which peer holds the advertisement. Peers interested in the advertisement will always get a copy from the publisher. With this method, if the peer that offers the service goes offline, the advertisement will become unavailable. That makes sense since, under that circumstance, the resource the advertisement announces will usually be unreachable nevertheless.

In *remote publication* the full advertisement is pushed via JXTA propagation to a set of peers. This method is useful in case that the advertisement must be reachable even when the publishing peer is offline (the resource is somehow 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 with a renewed expiration date. Advertisements may be periodically retransmitted in order to achieve permanency or update parameter changes.

2.2 Advertisement discovery

Advertisements can be located using the JXTA core Discovery Service via the Peer Discovery Protocol (PDP). This service is able to retrieve advertisements published both locally and remotely within a peer group, using two different approaches: by propagating the discovery query to rendezvous peers and the local subnet via IP multicast, or by forwarding the query to a specific peer.

In order to locate a specific advertisement, the discovery query defines a search criteria. The search criteria determines pairs of attribute-value that the desired advertisement should include. The Discovery Service may not provide a complete search on any available advertisement, but a "best effort" approach to deliver a selection of advertisements matching the search criteria. The selection returned may be random or predictable depending on the network configuration and no particular behavior should be assumed.

When a peer receives a discovery query, it checks if an advertisement matching the searching criteria is stored in its local cache. In that case, the peer directly sends an advertisement copy to the requester. In the case of a rendezvous peer, it searches in its SRDI distributed database the peer that holds a copy of the advertisement. Such peer is then notified by the rendezvous and will respond directly to the requesting peer. This advertisement retrieval mechanism is outlined in Figure 1.

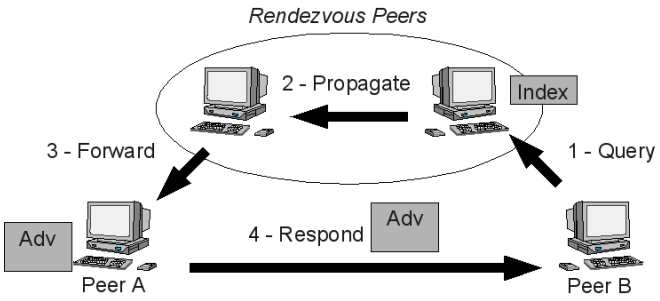

Fig. 1. Advertisement retrieval from Rendezvous peers

However, there are also two different scenarios in which a peer that requests an advertisement does not directly receive it even though the requested peer holds a proper copy. The first one is when it cannot be directly connected, usually because of Network Addres Translation (NAT) or a firewall. The second scenario is when the receiving endpoint is a minimal edge peer, a peer with limited resources. In such scenarios, advertisements are transmitted across the overlay network via intermediate peers which bounce the message up to the final destination. These peers are named relay peers in the former scenario and proxy peers in the latter one. In both cases, they can access and copy the advertisement content during transit.

2.3 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, since they are the gateway to network resources. As briefly exposed in Section 1, current advertisement security is mainly concerned with authenticity and non-repudiation, in order to prevent propagation of false advertisements and spoofing issues. An overview of the general state of advertisement security is provided in [15]. This section focuses on current security approaches to provide advertisement data privacy, a feature that can be used to control resource access or simply to distribute sensitive data over the network.

The current JXTA v2.0 specification [29] guarantees message privacy at transport level via its own implementation of generic Transport Layer Security (TLS) [31], providing private, mutually authenticated, reliable streaming communications. However, using this method, information security is only provided during transit by protecting the JXTA transport protocol at a lower layer. There is no method in order to specifically provide privacy to advertisements.

In the JXTA implementation of TLS, a virtual transport channel is created whenever a secure pipe is instantiated. This single channel accepts multi-

plexing data transmissions and is bidirectionally secured end-to-end, independently of JXTA relays and the underlying physical transports. However, such end-to-end security cannot be applied in some circumstances, as explained in subsection 2.2, because there is no direct communication between peers.

In the bidirectional TLS channel, both endpoints require X.509 [7] certificates in order to provide mutual authentication and to establish a private channel. Key authenticity for such certificates is provided by a common Trusted Third Party (TTP). When a peer joins a peer group, the group creator's root certificate is received protected by a TLS connection. The new member then uses a Certificate Signing Request (CSR) [20] to acquire a group membership certificate signed with the private key of the group creator. This information is stored locally in the peer's local cache under the protection of a passphrase. When a peer contacts another peer group member, they can be mutually authenticated using the TLS handshake's certificate request/response and certificate validation.

The TTP-based trust model needed for TLS is provided by the PSE Membership Service, becoming mandatory an environment where privacy is needed. 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, prove identity and provide a means for implementing access control in offered services. PSE is currently the only membership service where credentials take the form of X.509 certificate chains, providing the means for public key management in secure messaging.

3 Providing persistent advertisement encryption

We have reviewed current security solutions that are available to obtain some degree of privacy for the information contained in a JXTA advertisement. However, the existing solutions have the following constraints when providing data privacy in JXTA advertisements:

- Transient security: Advertisements are only secured during transport via TLS. Once they are stored into the local cache they are not encrypted anymore. Furthermore, the transient nature of the security layer also forces the application to decrypt all messages upon immediate reception, before encapsulation is discarded. However, most of the advertisements might never be used (its associated resource never accessed), the decryption process thus impacting system performance.
- Third party secure storage: With the existing proposals, any peer which

receives an advertisement to be stored in its local cache has full access to its content. It is not possible to allow peers to store advertisements while denying them access to some of its fields. This situation appears when advertisements are pushed to third parties in order to guarantee availability or redundancy, such as in remote publication or when the receiving peer is acting as rendezvous peer.

- All-or-nothing approach: Either peers have access to the full advertisement content or they cannot access it. Current mechanisms do not offer the possibility to provide partial access to advertisement content. This is very useful in order to limit access to specific services when a single advertisement is used to spread multiple services.
- Propagation not supported: The current TLS implementation in JXTA only provides point-to-point security but it does not support message propagation. However, support for this transport method would be desirable since it is the way advertisements are distributed in remote publication, an efficient method for bulk transfer.
- 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 trust model to one based on a TTP, and forcing the use of X.509 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. Furthermore, the use of a TTP inherits additional problems which increase the system's complexity, such as that of certificate chain management and revocation.

In our proposal, we solve all these constraints by defining an extension format for advertisements, based on XML encryption, that maintains the advertisement base structure as defined in the JXTA protocols specification. It is possible to choose which fields are secured and which peers may access each field, instead of being an all-or-nothing approach. By choosing which fields to secure, it is possible to control which peers may access specific services announced via the advertisement as well as avoid disrupting the discovery service standard operation.

Since privacy is at advertisement level, secured advertisements may be freely distributed and still remain encrypted when stored in other peers' local cache. Peers which do not support encryption may nevertheless process the clear text fields in a transparent way. Peers may freely choose which advertisements should be secured, and choose its own security degree, without impacting other peers. In addition, the use of a TTP is avoided by using Crypto-Based Identifiers (CBIDs). In this manner, JXTA applications which process sensitive data may provide some degree of privacy to specific advertisement exchanges [18,2].

3.1 Selective advertisement encryption

As it was pointed out in Section 2, JXTA advertisements are coded using XML format. Therefore, a logical approach to secure them is to use an XML encryption mechanism. There are several proposals to achieve selective encryption in XML documents [27,9,32]. We specifically use the standard *XMLenc* [32]. This approach ensures data privacy during transit and when the advertisement is stored in parties different from the one which generated the document, which is not supported in [27]. XMLenc also provides a better document size compared with [9], and it can be smoothly integrated with approaches which provide additional security services by using *XMLdsig* [33].

In our proposal, its is assumed that all peers within the peer group own a private/public key pair. Advertisements are selectively encrypted using a wrapped key encryption scheme (such as the one defined in [16]). For each advertisement field to be encrypted, a symmetric key is generated 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 one group member.

This scheme is applied to peer advertisements according to the profile shown in Figure 2. The advertisement's publisher may choose which advertisement fields will be encrypted. The encryption profile takes into account that all advertisements may contain three distinct types of fields, depending on the type of metadata stored within. Some constraints must be followed when selecting which fields are encrypted.

Mandatory fields must always exist in an advertisement, usually resource identifiers (such as the `GID` and `MSID` fields - see XML Listing 1). This fields are necessary in order to locate the advertisements via the Discovery Service and for that reason they should not be encrypted.

Optional fields provide additional information about the resource, but are not mandatory. They can be freely encrypted (or only some subset of them), but then the Discovery Service will not be able to index a resource using that field. Nevertheless, an encrypted field will not disrupt Discovery Service operations on unencrypted fields (encrypted fields will be ignored).

Service fields define specific services or resources and they are used to access them. If the service field is encrypted, only the selected subset of peers which may decrypt the data will be able to properly locate and access the resource. They usually take the form of `Svc` XML elements in Peer and Peer group Advertisements, containing Pipe Advertisements or Module Class Advertisements with executable code locations.

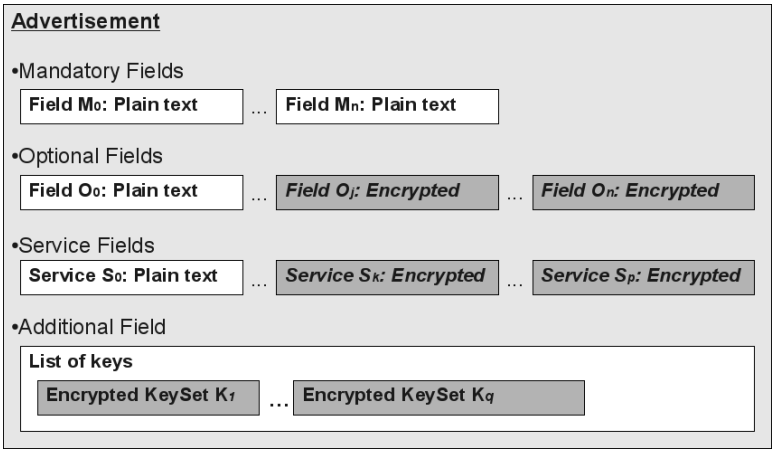

Fig. 2. Advertisement encryption schematics

Wrapped keys are included into the encrypted advertisement by introducing additional data, a list of keys. This data is encoded as a standard field named `KeyList`, following the same syntax as any other field in an advertisement. Such field contains an entry for each destination peer which should be able to access an encrypted field. Each encrypted field in the advertisement is associated to a set of a wrapped keys included in the `KeyList` field.

A wrapped key is defined by an XMLenc `EncryptedKey` element, which contains all cryptographic information necessary to decrypt such field. With this approach, XMLenc data can be added to advertisements without changing the basic elements in its schema definition, obtaining a transparent integration with JXTA and maintaining interoperability even when encrypted advertisements are used.

Encrypted fields within an advertisement are defined by XMLenc `EncryptedData` elements. Figure 3 shows the encryption profile we define in order to link each `EncryptedData` element to its corresponding `EncryptedKey`. Peers that receive the advertisement may identify which `EncryptedKey` fields may be decrypted with its own private key by searching for its Peer ID in the `KeyInfo` field of each `EncryptedKey` element. Every `EncryptedData` element may have several linked `EncryptedKey` elements (one for each peer which may access the service).

By using this XML profile and advertisement schematics, it is possible to accommodate selective field encryption as well as supporting advertisement propagation. In addition, keeping mandatory fields as plain text allows to achieve a trade off between data privacy and the capability to locate resources using the JXTA Discovery Service, since encrypted fields become invisible to the basic service operations. The detailed encryption/decryption process is described in subsection 3.4.

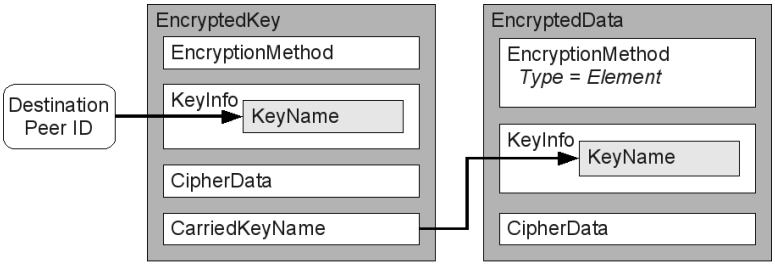

Fig. 3. XMLenc encryption profile

With this approach, apart from standard data privacy, it is now possible to provide access control to group services by encrypting specific advertisement fields: those which hold service definition and configuration parameters. Only those peers which can decrypt such fields will be able to ultimately locate the services. By using selective encryption, it is possible to choose which subset of peers may access each service.

3.2 Public key distribution and retrieval

The encryption of the selected field must be performed using each destination peer's public key and then such public keys have to be distributed. Key distribution is achieved directly via Peer Advertisements, by adding an additional `Svc` field to this advertisement type. With this approach, no extra message types or additional protocols are necessary, seamlessly integrating with the current capabilities of JXTA. In addition, those peers which do not support advertisement security do not need to process the additional field, but Peer Advertisements are still in a valid form.

A sample for the new Peer Advertisement format that enables the publication of key binding to a specific Peer ID is shown in Listing 2 (some identifiers have been shortened to improve readability).

The XMLdsig `KeyInfo` element is again used for key transport. However, in this case, it ultimately contains the encoded public key for the advertisement's peer instead of a reference (in contrast with encrypted advertisements, where a reference is used via the `KeyName` subelement). Several key types are supported by using different subelement types: `KeyValue` (raw public key, used in the example), `X509Data` (X.509 certificate), `PGPData` (PGP [26] key) and `SPKIData` (SPKI [8] certificates).

By supporting different key types, the system can integrate the field encryption method with the JXTA PSE environment (via a `X509Data` key transport). However, such integration is not mandatory and each application may use the public key model which most suits its needs.

XML Listing 2 - 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>
  <Svc>
    <MCID>urn:jxta:uuid-8C...05</MCID>
    <Parm>
    <ds:KeyInfo xmlns:ds="http://.../xmldsig#">
      <ds:KeyValue><ds:RSAKeyValue>
        <ds:Modulus>mHkQ5C6WU=</ds:Modulus>
        <ds:Exponent>AQAB</ds:Exponent>
      </ds:RSAKeyValue></ds:KeyValue>
    </ds:keyInfo>
    </Parm>
  </Svc>
</jxta:PA>
```

Using Peer Advertisements provides a way to manage keys in an easy and transparent manner by taking advantage of JXTA capabilities. Keys are pushed to other group members via the publication mechanisms explained in subsection 2.1. Key retrieval is accomplished by locating the Peer Advertisement via the PID element value using the Discovery Service as detailed in subsection 2.2.

3.3 Public key authenticity

Before a key may be used for encryption, its authenticity must be guaranteed. In our proposal, public key authenticity is achieved by using CBIDs, which avoids the intervention of a TTP. This concept was initially conceived for IPv6 addressing in order to solve the issue of address ownership, to avoid router supplantation attacks and bind update packet spoofing [30,17].

Under a CBID scenario, each peer's key pair is uniquely bound to its network identifier, providing key authenticity. In the absence of a TTP, this binding is created via applying a pseudo-random function to the public key and using the result (or part of it) as the peer identifier. There are other methods to bind key pairs to identifiers without the need of a TTP, such as an id-based approach [3] or self-certifying keys [19]. However, since JXTA IDs must follow a very specific format, they are not feasible.

A JXTA Peer ID should canonically, uniquely and unambiguously refer to a peer, being represented as 64 bytes according to the following fields: Peer Group UUID [22] (16 bytes long), Peer UUID (16 bytes long) and identifier type (1 byte long, its value being set at 03 for Peer ID's). The rest of bytes are unused. Notice that, according to this format, only 16 bytes, the Peer UUID, are really unique to each peer within the same peer group. For that reason, those are the only values that can be manipulated in order to generate CBIDs.

We generate CBIDs from the public key by applying the SHA-1 [21] hash algorithm on it. The result is then considered the Peer UUID in order to construct the full JXTA ID. Since SHA-1 produces 20 bytes but only 16 are needed, the first 4 most significant bytes are discarded. This process is summarized in Figure 4.

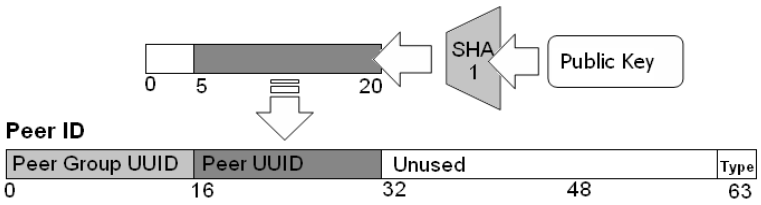

Fig. 4. CBID Generation

Using this method of CBID generation, no certificates are needed. In fact, 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 on the SHA-1 hash algorithm, it applies under the UUID type 5 specification [22]. This method also allows to integrate peers which use CBIDs with those who do not, since Peer IDs which are not CBIDs in a JXTA network are random numbers anyway.

It must be taken into account that a single ID is generated from a specific public key, so it could be a problem if two different public keys produce the same UUID field. The probability of such occurrence can be seen as a collision problem, similar to the Birthday Paradox [4]. The probability of a collision in such scenario is $p(n;d) \approx 1 - e^{-(n(n-1))/2d}$, where n is the number of values drawn from the distribution and $[1,d]$ the distribution range ($d = 2^{128}$ for a 16 bytes long, 128 bits, UUID). However, according to [23], a typical JXTA peer group should not have more than 32 members, since above this number performance starts to degrade. With 32 group members, the probability that two of them will have the same ID providing a different public key is about $1/20^{36}$, which is low enough.

In order to validate CBID ownership, whenever a peer advertisement is retrieved, the contained public key is extracted and then used to generate the source address, *i.e.* the CBID. If the obtained address is the same as the

claimed source address, stored in the `PID` field (as detailed in Listing 2), ownership is guaranteed.

3.4 Encryption and decryption process

The process of advertisement encryption that generates the XMLenc profile defined in previous sections can be described as follows.

Encryption:

(1) Each member of the peer group distributes its public key via its Peer Advertisement using the process detailed in subsection 3.2. Peer Advertisement publication is a standard procedure in JXTA.
(2) Peer A decides to publish some advertisement which must be secured.
(3) For each field (optional or service) defined within the advertisement, peer A chooses the subset of peers P_i, for $i = 1, \cdots, n$, that will be able to access it. Fields with the same subset of destination peers are then grouped into FG_j, for $j = 1, \cdots, m$, field groups.
(4) A new field, the `KeyList` field, is added to the advertisement.
(5) Each field group FG_j, for $j = 1, \cdots, m$, is processed in the following manner:
    (a) Both a random symmetric key K_j and an identifier id_j are generated by A.
    (b) Each field $F \in FG_j$ is encrypted according to XMLenc with K_j. Each original field becomes an XMLenc `EncryptedData` element. id_j is added to each `EncryptedData` element, as a `KeyInfo` element.
    (c) For each peer P_i, for $i = 1, \cdots, n$:
        (i) Peer A retrieves the peer advertisement of peer P_i using the core JXTA Discovery Service. Its public key PK_i is retrieved from the advertisement.
        (ii) Before encryption, A checks the authenticity of PK_i with the method described in subsection 3.3.
        (iii) K_j is wrapped (encrypted) using PK_i, generating an XMLenc `Encryptedkey` element. The `CarriedKeyName` field of this element is set to id_j. Its `KeyInfo` field is set to P_i's Peer ID by using a `KeyName` element as previously shown in figure 3.
        (iv) The `EncryptedKey` element, EK, is added to the `KeyList` field.
    (d) Once all peers in P_i, for $i = 1, \cdots, n$, have been processed, the `KeyList` field includes the wrapped keys for all peers P_i, for $i = 1, \cdots, n$.
(6) An encrypted advertisement has been generated according to the format defined in 3.1. For each encrypted field F, a set of wrapped keys exist

within the `KeyList` field which may decrypt it. Some fields may share the same set of wrapped keys.

(7) The advertisement is published via the core JXTA Discovery Service.

A sample encrypted Peer Group Advertisement after this process is shown in Listing 3 (some ID's and Base64 encoded data have been shortened in order to improve readability). It contains two original service fields (`Svc` elements), but only one of them, S_1, has been encrypted. The field entry in the encrypted advertisement corresponds to the `KeyList` field, which contains the wrapped keys necessary in order to decrypt S_1. In this example, only two peers (with Peer ID *urn:jxta:uuid-59...B6A403* and *urn:jxta:uuid-2B...D0A603*) may properly decrypt S_1. The identifier used to associate the encrypted field to the set of wrapped keys (encryption process, step 5a) is *KeyId-0*.

Decryption:

Whenever a peer $B = P_i$ for some $i = 1, \cdots, n$ wants to access to the resource:

(1) Peer B locates an advertisement via the core JXTA Discovery Service.
(2) Peer B locates the `KeyList` field within the advertisement.
(3) Peer B locates within the `KeyList` field the set of `EncryptedKey` elements, EK, which contain B's Peer ID in its `KeyInfo` field.
(4) For each `EncryptedKey`, $EncKey_i$, in EK:
   (a) Peer B selects the set of `EncryptedData` elements, ED, within the document which `KeyInfo` value match the `CarriedKeyName` value in $EncKey_i$. This value corresponds to the identifier id_j generated in step 5a of the encryption process.
   (b) Peer B's private key is used to decrypt the symmetric key, K_j, stored in $EncKey_i$.
   (c) Each `EncryptedData` element $EncData_i \in ED$ is decrypted using K_j.
   (d) The original field F has been recovered.
(5) Peer B obtains the final advertisement where some service entries may still be encrypted. B has no access to such entries, only to those he could satisfactorily decrypt.

It is worth mention that whenever a peer receives an encrypted advertisement, it can be stored into the peer's cache and its decryption can be deferred up to the moment the advertisement is actually used.

3.5 Encryption cost

The proposed security solution that offers privacy in JXTA advertisements implies a cost in message size since cryptographic information must be added.

XML Listing 3 - Selectively encrypted Peer Group Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<jxta:PGA xmlns:jxta="http://jxta.org" xml:space="preserve">
  <GID>urn:jxta:uuid-E7...02</GID>
  <MSID>urn:jxta:uuid-F2...06</MSID>
  <Name>My Peer Group Name</Name>
  <Desc>My Peer Group Description</Desc>
  <Svc>
    <xenc:EncryptedData xmlns:xenc="...xmlenc#" Type="...xmlenc#Content">
      <xenc:EncryptionMethod Algorithm="...xmlenc#aes128-cbc"/>
      <ds:KeyInfo xmlns:ds="...xmldsig#">
        <ds:KeyName>KeyId-0</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>iTXqXYoONtoxB44J...VUDg==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </Svc>
  <Svc>
    <MCID>urn:jxta:uuid-5326C14064F24A5DB634D14A1118ED0F05</MCID>
    <Parm type="myapp:MyServiceConfig">
      <MyParameter1>My Value 1</MyParameter1>
      <MyParameter2>My Value 2</MyParameter2>
    </Parm>
  </Svc>
  <KeyList>
    <xenc:EncryptedKey xmlns:xenc="...xmlenc#">
      <xenc:EncryptionMethod Algorithm="...xmlenc#rsa-1_5"/>
      <ds:KeyInfo xmlns:ds="...xmldsig#">
        <ds:KeyName>urn:jxta:uuid-59...B6A403</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>ip1y...QxI=</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:CarriedKeyName>KeyId-0</xenc: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-2B...D0A603</ds:KeyName>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>m9Jm...vLM=</xenc:CipherValue>
      </xenc:CipherData>
      <xenc:CarriedKeyName>KeyId-0</xenc:CarriedKeyName>
    </xenc:EncryptedKey>
  </KeyList>
</jxta:PGA>
```

The impact comes from two different element types (see Listing 3): `EncryptedKey` and `EncryptedData`. The first one is a completely new element (it does not appear in a standard advertisement) and it includes the keys needed to decrypt encrypted data. The `EncryptedData` element, although is also a new element, replaces a non-encrypted original element (a service field or an optional field) and then the complete size should not be considered as the security cost but only the difference between the plaintext version, $size_{of}$, and its encrypted form.

Regarding the `EncryptedKey` element, Table 1 shows its total size (in bytes) and the detailed information for each content element. Some elements have a fixed size and some others depend on different parameters. `EncryptedKey`

Content element	Fixed size	Variable size
EncryptedKey headers	86	-
EncryptionMethod	70	Wrap algorithm id ($RSA1.5 = 7$)
KeyInfo	175	-
CarriedKeyName	51	id_j length
CipherData	174	-
Total	**557**	

Table 1
`EncryptedKey` content element size (in bytes)

Content element	Fixed size	Variable size
EncryptedData header	138	-
EncryptionMethod	71	Algorithm id ($AES = 10$, $3DES = 13$)
KeyInfo	97	id_j length
CipherData	73	$AES \approx (18 + \frac{4}{3} size_{of})$ $3DES \approx (9 + \frac{4}{3} size_{of})$
Total	**379**	

Table 2
`EncryptedData` content element size (in bytes)

headers, `KeyInfo` and `CipherData` [1] are fixed size values while `CarriedKeyName` depends on the length of the id_j and `EncryptionMethod` depends on the wrap algorithm specification (for instance, 7 bytes are needed to encode the RSA1.5 algorithm identifier)

The `EncryptedData` element size is analyzed in Table 2. Data shows that the element size varies depending basically on the original field size $size_{of}$ although small differences between different encryption algorithms can also be observed. We detail the formula to calculate the final size using the Triple DES (3DES) [10] and AES [11] algorithms.

Just in order to provide some general information, for a 6 bytes id_j, a full key set for a typical group size (32 members) amounts to about a 18 Kbytes size increase due to key transport (the `KeyList` element). On the other hand, for each encrypted field, $419 + \frac{1}{3}(size_{of})$ bytes are added, where encryption with

[1] Assuming that an RSA public key of 1024 bits is used to wrap symmetric keys, the `CipherData` field size is fixed for different symmetric key sizes when such key values are below the RSA modulus.

AES is assumed.

4 Conclusions and further work

In this paper we have proposed a JXTA advertisement security scheme that allows to control which peers, within a peer group, can access available resources.

The main contributions are the following ones. First of all, the proposed method provides advertisement privacy in a lightweight manner, without the need of a TTP-based trust model, since CBIDs are used. This method can be adapted to a broad choice of key transport specifications. Furthermore, it maintains end-to-end advertisement security for its whole lifetime, not just during transport, by encrypting data at the application layer, keeping it secure even when stored in a peer's local cache. Even though JXTA may use its current implementation of TLS in order to provide a message security layer, such security is transient. Using our approach, it is also possible to provide privacy in advertisement propagation, which is not currently available in JXTA.

In addition, our proposed method also takes special care to keep advertisement interoperability by maintaining as much as possible of the base advertisement format in the encrypted advertisement format, public key wrapping and key distribution, instead of creating completely new metadata documents.

Encrypted advertisement interoperability is achieved by selectively encrypting advertisement fields. Mandatory fields can still be accessed and processed by the Discovery Service (both locally or at the SRDI super-network) in order to search resources within the network. Key wrapping and distribution may be processed by JXTA in a transparent way by using the possibilities of the advertisements' service field definition.

Another benefit of maintaining the advertisements base format is the possibility to defer decryption until the moment the advertisement has to be used, or even allowing peers to store secure advertisements which will never be used (for example, stored to provide redundancy via remote publication). This feature may improve the peer's performance since there may be a lot of advertisement traffic in a given network that the peer may not ever use.

We have assessed the cost of using our proposal from an advertisement size increase standpoint. From this analysis, we see that most of the data length overhead is moslty related to wrapped key distribution. However, such overhead remains fairly static, since symmetric key length has little efect on the `EncryptedKey` fields. For an RSA 1.5 key wrapping and a 6 byte long key

identifier, each distributed key amounts to a size increase of about 570 bytes. The most variable overhead is the one directly related to encrytped field data length.

Selective encryption also allows to achieve published service access control by moving away from the current all-or-nothing approach. Peers have the possibility to define which sets of group members may access specific data within an advertisement. Using this method, it is possible to provide discretionary access control to published resources as well as data privacy.

Currently, this proposal mechanism has been specified an implemented under the approved SUN project JXTA-XMLsec [13], and libraries can also be downloaded from the project's homepage [2]. This prototype is under testing and improvement by its integration into the JXTA-Overlay framework [24]. The main goal of this process is providing the JXTA-Overlay framework primitives and functions with an acceptable level of security. Further work also goes towards integrating the proposal with XML signature based advertisements [14], providing a full security suite for JXTA advertisement publication and discovery.

References

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

[2] A. Sanna, C. Zunino, L. Ciminiera, A distributed JXTA-based architecture for searching and retrieving solar data, Future Generation Computer Systems, 21 (3) pp. 349–359 (2005)

[3] A. Shamir, Identity-based cryptosystems and signature schemes, In proceedings of Advances in Cryptology (CRYPTO '84), pp. 47–53 (1984)

[4] B. Schneier, Applied Cryptography. John Wiley and Sons (1996)

[5] B. Traversat, A. Arora et al, Project JXTA 2.0 super-peer virtual network, Technical report, Sun Microsystems Inc. (2003).

[6] B. Traversat, M. Abdelaziz and E. Pouyou, Project JXTA: A loosely-consistent dht rendezvous walker, Technical report, Sun Microsystems Inc. (2003).

[7] CCITT, The directory authentication framework. Recommendation (1988)

[8] C. Ellison, B. Frantz et al, SPKI certificate theory (1999)

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

[2] http://kison.uoc.edu, "Research"

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

[11] FIPS - Federal Information Processing Standard, Advanced Encryption Standard (AES), National Bureau of Standards, U.S. Department of Commerce (2001)

[12] Gnutella, `http://rfc-gnutella.sourceforge.net` (2000)

[13] J. Arnedo-Moreno, JXTA-XMLsec Project Homepage, `https://jxta-xmlsec.dev.java.net`, developer account is required (2008)

[14] J. Arnedo-Moreno, J. Herrera-Joancomartí, Persistent interoperable security for jxta advertisements, In proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC'08), pp. 354 – 359 (2008)

[15] J. Arnedo-Moreno, J. Herrera-Joancomartí, A survey on security in JXTA applications, Journal on Systems and Software. To be published (accepted, in press) (2009)

[16] J. Staddon, B. Kaliski, PKCS #1: RSA cryptography specifications. Version 2.0, `http://www.ietf.org/rfc/rfc2437.txt` (1998)

[17] L. Bononi, C. Tacconi,, Intrusion detection for secure clustering and routing in mobile multi-hop wireless networks, International Journal on Information Security, vol. 6, no. 6, 379–392 (2007)

[18] M. Cannataro, D. Talia, G. Tradigo, P. Trunfio, P. Veltri, SIGMCC: A system for sharing meta patient records in a Peer-to-Peer environment, Future Generation Computer Systems, 24 (3) pp. 222–234 (2008)

[19] M. Girault, Self-certified public keys, In proceedings of Advances in Cryptology (EUROCRYPT '91), pp. 490–497 (1991)

[20] M. Nystrom, B. Kaliski, Certification request syntax specification, `http://www.ietf.org/rfc/rfc2986.txt` (2000)

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

[22] P. Leach, M. Mealling, R. Salz, RFC 1422: A universally unique identifier (UUID) URN namespace (2005)

[23] R. Deters, E. Halepovic, The JXTA performance model and evaluation, Future Generation of Computer Systems 21 (3) pp. 377–390 (2005)

[24] R. Fernandez, F. Xhafa, The JXTA-Overlay Project, `https://jxta-overlay.dev.java.net/` (2007)

[25] R. Moreno-Vozmediano, A hybrid mechanism for resource/service discovery in ad-hoc grids, Future Generation of Computer Systems 25 (7) pp. 717–727 (2009)

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

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

[28] SUN Microsystems, Project JXTA, http://www.jxta.org (2001)

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

[30] T. Aura, RFC 3972: Cryptographically Generated Adresses (CGA), http://www.ietf.org/rfc/rfc3972.txt (2005)

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

[32] W3C, XML encryption syntax and processing, http://www.w3.org/TR/xmlenc-core/ (2002)

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