XEP-0290: Encapsulated Digital Signatures in XMPP

Kurt Zeilenga
mailto:Kurt.Zeilenga@Isode.COM
xmpp:Kurt.Zeilenga@Isode.COM

2011-01-28
Version 0.2

Status	Type	Short Name
Deferred	Standards Track	N/A

This document provides a technical specification for Encapsulated Digital Signatures in the Extensible Messaging and Presence Protocol (XMPP).

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

This document is one of two proposals for digital signatures in XMPP. It is expected that only one of these proposals be progressed beyond Experimental on the Standards Track.

This document provides a technical specification for Encapsulated Digital Signatures in Extensible Messaging and Presence Protocol (XMPP [1]).

XMPP Digital Signatures may be used to provide signer authentication, data integrity, non-repudiation, and other security services.

This extension is intended to be highly flexible, supporting a wide range of applications. The extension not only supports signing by the originator, but by other entities which handle XMPP stanzas. Multiple entities may independently sign a stanza.

A signed manifest approach is used to allow selective signing (only select elements may be included in the manifest) and to allow flexibility in handling verification errors.

This extension is intended to support *optimistic signing*.

This document offers an encapsulated signature approach based upon XML Signature [2] (XMLDSIG). Implementations of this extension not required to fully implement the XML DSIG specification, they may implement only the minimal subset necessary to support this extension.

It is noted that a number of object-level XMPP digital signature extensions have been specified over the years. These include RFC 3923 [3] (XMPP E2E), Encrypted Session Negotiation (XEP-0116) [4] (XMPP PGP), and Encapsulating Digital Signatures in XMPP (XEP-0285) [5]. The limited applicability of encapsulating signature approaches in XMPP is discussed in Design Considerations for Digital Signatures in XMPP (XEP-0274) [6].

2 Signing XMPP Stanzas

The process that a sending agent follows for securing stanzas is very similar regardless of the form of stanza (i.e., <iq/>, <message/>, or <presence/>).

The signer begins with the cleartext version of the <message/> stanza "S":

```
<message from='juliet@capulet.net/balcony'
         id='183ef129'
         to='romeo@montague.net'>
    <thread>8996aef0-061d-012d-347a-549a200771aa</thread>
    <body>Wherefore art thou, Romeo?</body>
```

[1] Extensible Messaging and Presence Protocol (XMPP) <https://xmpp.org/>.

[2] XML Signature Syntax and Processing <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>.

[3] RFC 3923: End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP) <http://tools.ietf.org/html/rfc3923>.

[4] XEP-0116: Encrypted Session Negotiation <https://xmpp.org/extensions/xep-0116.html>.

[5] XEP-0285: Encapsulating Digital Signatures in XMPP <https://xmpp.org/extensions/xep-0285.html>.

[6] XEP-0274: Design Considerations for Digital Signatures in XMPP <https://xmpp.org/extensions/xep-0274.html>.

```
</message>
```

This is modified prior to signing as follows:

- Default attribute values are added. (Namespace declarations are not modified.)
- Each child element of the stanza is augmented by a id attribute qualified in the urn:xmpp:dsig:0 namespace. As these attributes are used to identify the element within a manifest, they must be sufficient unique.

```
<message from='juliet@capulet.net'
         id='183ef129'
         to='romeo@montague.net'
         type='chat'>
    <thread xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-1">8996aef0-061d-012d
        -347a-549a200771aa</thread>
    <body xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-2">Wherefore art thou,
        Romeo?</body>
</message>
```

The signer builds a stanza description object containing a signer element, the stanza element, and a timestamp element. The signer element value is the bare JID of the signing entity. When the signing entity is a service entity, the JID may only contain service domain. The stanza-desc element is the stanza prepared above with each of its child elements replaced by an reference element. The reference element references the child element via a "urn:xmpp:dsig:ref:0" URI with an anchor of the value of child element's d:id. The value of the reference element is the digest value produced by the digest method after the specified transforms are applied.

```
<stanza-desc id="stanza-desc" xmlns="urn:xmpp:dsig:0">
    <signer>juliet@capulet.net</signer>
    <Transforms><Transform Algorithm="urn:xmpp:dsig:transform:0"/></
        Transform>
    <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <message from='juliet@capulet.net'
             id='183ef129'
             to='romeo@montague.net'
             type='chat'>
        <reference URI='urn:xmpp:dsig:ref:0#xxxx-1'>...</reference>
        <reference URI='urn:xmpp:dsig:ref:0#xxxx-2'>...</reference>
    </message>
    <timestamp>2010-11-11T13:33:00.123Z</timestamp>
</stanza-desc>
```

The signer then builds a SignedInfo element.

```
<SignedInfo>
```

```
    <CanonicalizationMethod Algorithm="urn:xmpp:dsig:transform:0"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-
        sha1"/
    <Reference URI="#stanza-desc">
        <Transforms>
            <Transform Algorithm="http://www.w3.org/2006/12/xml-c14n11
                "/>
        </Transform>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
            sha1"/>
        <DigestValue>...</DigestValue>
    </Reference>
</SignedInfo>
```

And then produces a Signature element:

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2006/12/
            xml-c14n11"/>
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
            dsa-sha1"/
        <Reference URI="#stanza-desc">
            <Transforms>
                <Transform Algorithm="http://www.w3.org/2006/12/xml-
                    c14n11"/>
            </Transform>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig
                #sha1"/>
            <DigestValue>...</DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue/>
    <Object>
        <stanza-desc id="stanza-desc" xmlns="urn:xmpp:dsig:0">
            <signer>juliet@capulet.net</signer>
            <Transforms><Transform Algorithm="
                urn:xmpp:dsig:transform:0"/></Transform>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig
                #sha1"/>
            <message from='juliet@capulet.net'
                     id='183ef129'
                     to='romeo@montague.net'
                     type='chat'>
                <reference URI='urn:xmpp:dsig:ref:0#xxxx-1'>...</
                    reference>
                <reference URI='urn:xmpp:dsig:ref:0#xxxx-2'>...</
                    reference>
            </message>
```

```
            <timestamp>2010-11-11T13:33:00.123Z</timestamp>
        </stanza-desc>
    </Object>
</Signature>
```

The signer than computes the SignatureValue element, processing the Signature element as a detached signature, and replaces the empty Signature element with it. Finally, the signer inserts the Signature element into stanza and forwards the stanza as it normally would.

```
<message from='juliet@capulet.net'
         id='183ef129'
         to='romeo@montague.net'
         type='chat'>
    <thread xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-1">8996aef0-061d-012d
        -347a-549a200771aa</thread>
    <body xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-2">Wherefore art thou,
        Romeo?</body>
    <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
            <CanonicalizationMethod Algorithm="http://www.w3.org
                /2006/12/xml-c14n11"/>
            <SignatureMethod Algorithm="http://www.w3.org/2000/09/
                xmldsig#dsa-sha1"/
            <Reference URI="#stanza-desc">
                <Transforms>
                    <Transform Algorithm="http://www.w3.org/2006/12/
                        xml-c14n11"/>
                </Transform>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/
                    xmldsig#sha1"/>
                <DigestValue>...</DigestValue>
            </Reference>
        </SignedInfo>
        <SignatureValue>...</SignatureValue>
        <Object>
            <stanza-desc id="stanza-desc" xmlns="urn:xmpp:dsig:0">
                <signer>juliet@capulet.net</signer>
                <Transforms><Transform Algorithm="
                    urn:xmpp:dsig:transform:0"/></Transform>
                <DigestMethod Algorithm="http://www.w3.org/2000/09/
                    xmldsig#sha1"/>
                <message from='juliet@capulet.net'
                    id='183ef129'
                    to='romeo@montague.net'
                    type='chat'>
                    <reference URI='urn:xmpp:dsig:ref:0#xxxx-1'>...</
                        reference>
                    <reference URI='urn:xmpp:dsig:ref:0#xxxx-2'>...</
                        reference>
```

```
                </message>
                <timestamp>2010-11-11T13:33:00.123Z</timestamp>
            </stanza-desc>
        </Object>
    </Signature>
</message>
```

3 Transformation Algorithm

For referenced elements of a stanza, the referenced element is normalized (omitting comments), as detailed in Canonical XML [7], as a document subset of a document containing a stream element containing the appropriate jabber:client stanza element containing the stanza with the referenced elements. Note that jabber:client stanzas are used when constructing this document even when a server is signing a stanza to be sent to another server.
For the stanza in Example 2, the input document would be:

```
<stream:stream xmlns='jabber:client" xmlns:stream='http://etherx.
    jabber.org/streams'>
    <message from='juliet@capulet.net'
             id='183ef129'
             to='romeo@montague.net'
             type='chat'>
        <thread xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-1">8996aef0-061d
    -012d-347a-549a200771aa</thread>
        <body xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-2">Wherefore art
    thou, Romeo?</body>
    </message>
</stream:stream>
```

The canonical form of element referenced by urn:xmpp:dsig:ref:0#xxx-1 would be:

```
<thread xmlns="jabber:client" xmlns:d="urn:xmpp:dsig:0" d:id="xxxx-1">
    8996aef0-061d-012d-347a-549a200771aa</thread>
```

4 Stanza Element References

The URI 'uri:xmpp:dsig:ref:0#xxxx" refers to the child element of the stanza which contains the 'uri:xmpp:dsig:0' 'id' attribute with the value "xxxx".

[7] Canonical XML 1.0 <http://www.w3.org/TR/xml-c14n>.

5 Inclusion and Checking of Timestamps

Timestamps are included to help prevent replay attacks. All timestamps MUST conform to DATETIME [8] and be presented as UTC with no offset, always including the seconds and fractions of a second to three digits (resulting in a datetime 24 characters in length). Absent a local adjustment to the sending agent's perceived time or the underlying clock time, the sending agent MUST ensure that the timestamps it sends to the receiver increase monotonically (if necessary by incrementing the seconds fraction in the timestamp if the clock returns the same time for multiple requests). The following rules apply to the receiving application:

- It MUST verify that the timestamp received is within five minutes of the current time, except as described below for offline messages.
- If the foregoing check fails, the timestamp SHOULD be presented to the receiving entity (human or application) marked with descriptive text indicating "old timestamp" or "future timestamp" and the receiving entity MAY return a stanza error to the sender (except as precluded in the protocol).

The foregoing timestamp checks assume that the recipient is online when the message is received. However, if the recipient is offline then the server will probably store the message for delivery when the recipient is next online (offline storage does not apply to <iq/> or <presence/> stanzas, only <message/> stanzas). As described in Best Practices for Handling Offline Messages (XEP-0160) [9], when sending an offline message to the recipient, the server SHOULD include delayed delivery data as specified in Delayed Delivery (XEP-0203) [10] so that the recipient knows that this is an offline message and also knows the original time of receipt at the server. In this case, the recipient SHOULD verify that the timestamp received in the encrypted message is within five minutes of the time stamped by the recipient's server in the <delay/> element.

6 Mandatory-to-Implement Cryptographic Algorithms

All implementations MUST support the following algorithms. Implementations MAY support other algorithms as well.

- TBD

7 Certificates

To participate in end-to-end signing using the methods defined in this document, a client needs to possess an X.509 certificate. It is expected that many clients will generate their

[8] RFC 3339: Date and Time on the Internet: Timestamps <http://tools.ietf.org/html/rfc3339>.
[9] XEP-0160: Best Practices for Handling Offline Messages <https://xmpp.org/extensions/xep-0160.html>.
[10] XEP-0203: Delayed Delivery <https://xmpp.org/extensions/xep-0203.html>.

own (self-signed) certificates rather than obtain a certificate issued by a certification authority (CA). In any case the certificate MUST include an XMPP address that is represented using the ASN.1 Object Identifier "id-on-xmppAddr" as specified in Section 5.1.1 of RFC 3920bis.

8 Security Considerations

TBD.

9 XMPP Registrar Considerations

9.1 XML Namespace Name for Signed Data in XMPP

A URN sub-namespace of signed content for the Extensible Messaging and Presence Protocol (XMPP) is defined as follows.

URI: urn:xmpp:dsig

Specification: ProtoXEP

Description: This is an XML namespace name of signed content for the Extensible Messaging and Presence Protocol as defined by ProtoXEP.

Registrant Contact: XSF