Network Working Group
Internet-Draft
Expires: February 20, 2004

P. Saint-Andre
Jabber Software Foundation

August 22, 2003

End-to-End Object Encryption in XMPP
draft-ietf-xmpp-e2e-05

Status of this Memo

Copyright Notice

Abstract

This document defines a method for end-to-end object signing and encryption in the Extensible Messaging and Presence Protocol (XMPP).

Table of Contents

1. Introduction

This document define a method for end-to-end signing and encryption in the Extensible Messaging and Presence Protocol (XMPP). (For information about XMPP, see XMPP Core [1] and XMPP IM [2].) The method defined herein enables a sender to encrypt and/or sign an instant message sent to a specific recipient, encrypt and/or sign presence information that is directed to a specific user, and sign presence information that is broadcasted to a specific user. This document thereby helps the XMPP specifications meet the requirements defined in RFC 2779 [3].

1.1 Terminology

This document inherits terminology defined in RFC 2633 [4], RFC 2778 [5], RFC 3369 [6], and XMPP Core [1].

The capitalized key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [7].

1.2 Discussion Venue

The authors welcome discussion and comments related to the topics presented in this document. The preferred forum is the <xmppwg@jabber.org> mailing list, for which archives and subscription information are available at <http://www.jabber.org/cgi-bin/mailman/listinfo/xmppwg/>.

1.3 Intellectual Property Notice

This document is in full compliance with all provisions of Section 10 of RFC 2026. Parts of this specification use the term "jabber" for identifying namespaces and other protocol syntax. Jabber[tm] is a registered trademark of Jabber, Inc. Jabber, Inc. grants permission to the IETF for use of the Jabber trademark in association with this specification and its successors, if any.

2. Requirements

For the purposes of this document, we stipulate the following requirements:

1. The method defined MUST address encryption and signing requirements for minimal instant messaging and presence only, as those are defined in RFC 2779 [3]. The method is NOT REQUIRED to support non-IM applications of XMPP, nor to support advanced instant messaging and presence functionality that is outside the scope of RFC 2799. In particular, the method MUST address the following requirements defined in RFC 2779:

   * The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT MESSAGE) has not been corrupted or tampered with. (Section 2.5.1)

   * The protocol MUST provide means to ensure confidence that a received message (NOTIFICATION or INSTANT MESSAGE) has not been recorded and played back by an adversary. (Section 2.5.2)

   * The protocol MUST provide means to ensure that a sent message (NOTIFICATION or INSTANT MESSAGE) is only readable by ENTITIES that the sender allows. (Section 2.5.3)

   * The protocol MUST allow any client to use the means to ensure non-corruption, non-playback, and privacy, but the protocol MUST NOT require that all clients use these means at all times. (Section 2.5.4)

   * When A establishes a SUBSCRIPTION to B's PRESENCE INFORMATION, the protocol MUST provide A means of verifying the accurate receipt of the content B chooses to disclose to A. (Section 5.1.4)

   * The protocol MUST provide A means of verifying that the presence information is accurate, as sent by B. (Section 5.3.1)

   * The protocol MUST provide A means of ensuring that no other PRINCIPAL C can see the content of M. (Section 5.4.6)

   * The protocol MUST provide A means of ensuring that no other PRINCIPAL C can tamper with M, and B means to verify that no tampering has occurred. (Section 5.4.7)

2. The method defined MUST enable interoperability with non-XMPP messaging systems that support the Common Presence and Instant

Messaging (CPIM) specifications defined by the Instant Messaging and Presence (IMPP) Working Group. Therefore:

* Prior to encrypting or signing, the format of an instant message must conform to the CPIM Message Format defined in MSGFMT [8].

* Prior to encrypting or signing, the format of presence information must conform to the CPP Presence Information Data Format defined in PIDF [9].

3. The method MUST follow the required procedures (including the specific algorithms) defined in Common Profile for Instant Messaging [10] and Common Profile for Presence [11]. In particular, these documents specify:

   * Encryption MUST use S/MIME [4] encryption with CMS [6] EnvelopeData.

   * Signing MUST use S/MIME [4] signatures with CMS [6] SignedData.

4. In order to enable interoperable implementations, sending and receiving applications MUST implement the algorithms defined under Section 6.7.

3. Securing Messages

In order to encrypt a message, a sending entity MUST use the following procedure:

1. Generate a "Message/CPIM" object as defined in MSGFMT [8].

2. Encrypt and/or sign both the headers and content of the "Message/CPIM" object as specified in Requirement 3 of Section 2 above.

3. Provide the resulting multipart S/MIME object (see RFC 1847 [12]) as the CDATA of an <e2e/> child of a <message/> stanza, with the <e2e/> element scoped by the 'urn:ietf:params:xml:ns:xmpp-e2e' namespace (note that this namespace name adheres to the format defined in The IANA XML Registry [13]).

Example 1: Sender generates "Message/CPIM" object:

```
Content-type: Message/CPIM

From: Juliet Capulet <im:juliet@example.com>
To: Romeo Montague <im:romeo@example.net>
DateTime: 2003-05-14T11:45:36Z
Subject: Imploring

Content-type: text/plain; charset=utf-8
Content-ID: <1234567890@example.com>

Wherefore art thou, Romeo?
```

Example 2: Sender generates signed message (the 'from' address on the XMPP message stanza is stamped by sender's server):

```
<message to='romeo@example.net/orchard' type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
<![CDATA[
Content-Type: multipart/signed; boundary=next;
              micalg=sha1;
              protocol=application/pkcs7-signature

--next
Content-type: Message/CPIM

From: Juliet Capulet <im:juliet@example.com>
To: Romeo Montague <im:romeo@example.net>
DateTime: 2003-05-14T23:45:36Z
Subject: Imploring
```

```
   Content-type: text/plain; charset=utf-8
   Content-ID: <1234567890@example.com>

   Wherefore art thou, Romeo?
   --next
   Content-Type: application/pkcs7-signature

   [signed body part]

   --next--
   ]]>
     </e2e>
   </message>
```

4. Securing Presence

In order to encrypt presence information, a sending entity MUST use the following procedure:

1. Generate an "application/pidf+xml" object as defined in PIDF [9].

2. Encrypt and/or sign the "application/pidf+xml" object as specified in Requirement 3 of Section 2 above.

3. Provide the resulting S/MIME object as the CDATA of an <e2e/> child of a <presence/> stanza, with the <e2e/> element scoped by the 'urn:ietf:params:xml:ns:xmpp-e2e' namespace (note that this namespace name adheres to the format defined in The IANA XML Registry [13]). The <presence/> stanza MUST include a 'to' attribute, i.e., it must be an instance of directed presence as defined in XMPP IM [2].

Example 3: Sender generates "application/pidf+xml" object:

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:im="urn:ietf:params:xml:ns:pidf:im"
          entity="pres:juliet@example.com">
  <tuple id="h40zny"
    <status>
      <basic>open</basic>
      <im:im>away</im:im>
    </status>
    <note xml:lang="en">retired to the chamber</note>
    <timestamp>2003-05-14T23:53:11Z</timestamp>
  </tuple>
</presence>
```

Example 4: Sender generates signed presence (the 'from' address on the XMPP presence stanza is stamped by sender's server):

```
<presence to='romeo@example.net/orchard'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
<![CDATA[
Content-Type: multipart/signed; boundary=next;
              micalg=sha1;
              protocol=application/pkcs7-signature

--next
Content-type: application/pidf+xml
Content-ID: <2345678901@example.com>
```

```
<xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
          xmlns:im="urn:ietf:params:xml:ns:pidf:im"
          entity="pres:juliet@example.com">
  <tuple id="hr0zny">
    <status>
      <basic>open</basic>
      <im:im>away</im:im>
    </status>
    <note xml:lang="en">retired to the chamber</note>
    <timestamp>2003-05-14T23:53:11Z</timestamp>
  </tuple>
</presence>
--next
Content-Type: application/pkcs7-signature

[signed body part]

--next--
]]>
  </e2e>
</presence>
```

5. Securing Arbitrary XMPP Data

The foregoing sections of this document describe how to secure "least common denominator" messaging and presence data of the kind that can be directly translated into the MSGFMT or PIDF formats. However, XMPP possesses a third base-level stanza type (<iq/>) in addition to <message/> and <presence/>, as well as the ability to include extended XML data within arbitrary child elements of the three core stanza types. Therefore it would be desirable to secure such data if possible.

Because MSGFMT [8] specifies the ability to encapsulate any MIME type, the approach taken in this document is to include arbitrary XMPP data in a new MIME type, "application/xmpp+xml". The root element for this MIME type is <xmpp/>, and the root element MUST contain one and only one child element, corresponding to one of the XMPP stanza types (i.e., message, presence, or iq) if the default namespace is 'jabber:client' or 'jabber:server' as defined in XMPP Core [1].

The following examples illustrate the structure of the "application/xmpp+xml" MIME type.

Example 5: Message stanza with extended data contained in "application/xmpp+xml" MIME type:

```
<?xml version='1.0' encoding='UTF-8'?>
<xmpp xmlns='jabber:client'>
  <message
      from='iago@example.com/pda'
      to='emilia@example.com/cell'>
    <body>
      I told him what I thought, and told no more
      Than what he found himself was apt and true.
    </body>
    <evil xmlns='http://jabber.org/protocol/evil'/>
  </message>
</xmpp>
```

Example 6: Presence stanza with extended data contained in "application/xmpp+xml" MIME type:

```
<?xml version='1.0' encoding='UTF-8'?>
<xmpp xmlns='jabber:client'>
  <presence from='iago@example.com/pda'>
    <show>dnd</show>
    <status>Fomenting dissension</status>
    <evil xmlns='http://jabber.org/protocol/evil'/>
```

```
  </presence>
</xmpp>
```

Example 7: IQ stanza with extended data contained in "application/xmpp+xml" MIME type:

```
<?xml version='1.0' encoding='UTF-8'?>
<xmpp xmlns='jabber:client'>
  <iq type='result'
      from='iago@example.com/pda'
      to='emilia@example.com/cell'
      id='evil1'>
    <query xmlns='jabber:iq:version'>
      <name>Stabber</name>
      <version>666</version>
      <os>FiendOS</os>
    </query>
    <evil xmlns='http://jabber.org/protocol/evil'/>
  </iq>
</xmpp>
```

6. Rules for S/MIME Generation and Handling

6.1 Certificate Enrollment

S/MIME v3 does not specify how to obtain a certificate from a certificate authority, but instead mandates that every sending agent must already have a certificate. The PKIX Working Group has, at the time of this writing, produced two separate standards for certificate enrollment: CMP (RFC 2510) and CMC (RFC 2792). Which method to use for certificate enrollment is outside the scope of this document.

6.2 Certificate Retrieval

A receiving agent MUST provide some certificate retrieval mechanism in order to gain access to certificates for recipients of digital envelopes. This document does not cover how S/MIME agents handle certificates, only what they do after a certificate has been validated or rejected. S/MIME certification issues are covered in RFC 2632 [14].

At a minimum, for initial S/MIME deployment, a user agent could automatically generate a message to an intended recipient requesting that recipient's certificate in a signed return message. Receiving and sending agents SHOULD also provide a mechanism to allow a user to "store and protect" certificates for correspondents in such a way so as to guarantee their later retrieval.

6.3 Certificate Names

End-entity certificates used in the context of this document SHOULD a valid instant messaging address. The address SHOULD be one of the following:

1. An "instant inbox address" as defined in RFC 2778 [5] and MSGFMT [8]. As explained in XMPP CPIM Mapping [16], an instant inbox address maps to a "bare JID" (XMPP <node@domain>) once the 'im:' URI scheme has been removed. The appropriate container for instant inbox address shall be defined in MSGFMT [8].

2. An XMPP address (JID) as defined in XMPP Core [1]; the address should be of the form <node@domain> (i.e., a "bare JID"), although any valid JID form MAY be used. The JID SHOULD be contained in the subjectAltName extension, and SHOULD NOT be in the subject distinguished name.

The value of the JID contained in the XMPP 'from' attribute SHOULD match the JID provided in the signer's certificate, with the exception that the resource identifier portion of the JID contained

in the 'from' attribute MAY be ignored for matching purposes.

Receiving agents MUST recognize XMPP addresses (JIDs) in the subjectAltName field.

Receiving agents SHOULD check that sending JID matches a JID provided in the signer's certificate, with the exception that the resource identifier portion of the JID contained in the 'from' attribute MAY be ignored for matching purposes. A receiving agent SHOULD provide some explicit alternate processing of the message if this comparison fails, which may be to display a message that shows the recipient the addresses in the certificate or other certificate details.

The subject alternative name extension is used in S/MIME as the preferred means to convey the JID that corresponds to the entity for this certificate. Any JIDs present SHOULD be encoded using the otherName CHOICE of the subjectAltName type, where the type-id is "xmpp" and the value is the bare JID of the entity.

6.4 Transfer Encoding

According to various S/MIME specifications for message wrapping, CMS objects MAY optionally be wrapped in MIME to dynamically support 7-bit transport. Because it is expected that XMPP will not be used to interface with older 7-bit systems, this outer wrapping is NOT REQUIRED for XMPP transport, and generally SHOULD NOT be applied in a homogeneous XMPP environment or in an environment that supports XMPP-CPIM gateways.

6.5 Attachment of Signatures

Sending agents SHOULD attach a signature to each encrypted message or presence stanza, but are NOT REQUIRED to do so.

6.6 Inclusion of Certificates

Sending agents are NOT REQUIRED to include the sender's certificate along with each encrypted message or presence stanza.

6.7 Mandatory to Implement Technologies

At a minimum, all implementations MUST support the following CMS algorithms as defined in RFC 3370 [15]:

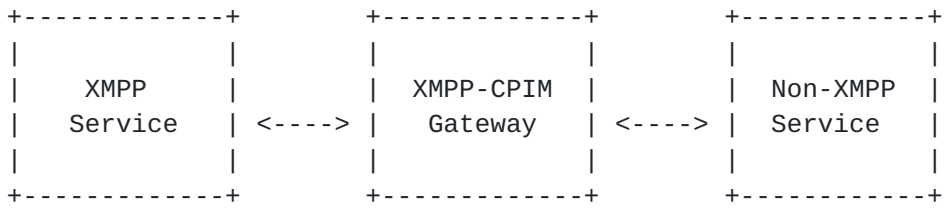
for digest: DIGEST-MD5

for signing: RSA

for content encryption: Triple-DES CBC

7. Secure Communications Through a Gateway

A common method for achieving interoperability between two disparate services is through the use of a "gateway" that interprets the protocols of each service and translates them into the protocols of the other. The CPIM specifications (specifically MSGFMT [8] and PIDF [9] define the common profiles to be used for interoperability between instant messaging and presence services that comply with RFC 2779 [3]. In the case of communications between an XMPP service and a non-XMPP service, we can visualize this relationship as follows:



The end-to-end encryption method defined herein enables the exchange of encrypted and/or signed instant messages and presence through an XMPP-CPIM gateway. In particular:

- When a gateway receives a secured XMPP message or presence stanza from the XMPP service that is addressed to a user on the non-XMPP service, it MUST remove the XMPP "wrapper" (everything down to and including the <e2e> and </e2e> tags) in order to reveal the multipart S/MIME object, then route the object to the non-XMPP service (first wrapping it in the protocol used by the non-XMPP service if necessary).

- When a gateway receives a secured non-XMPP instant message or presence document from the non-XMPP service that is addressed to a user on the XMPP service, it MUST remove the non-XMPP "wrapper" (if any) in order to reveal the multipart S/MIME object, wrap the object in an XMPP message or presence "wrapper" (including the <e2e> and </e2e> tags), and then route the XMPP stanza to the XMPP service.

The wrapped S/MIME object MUST be immutable and MUST NOT be modified by an XMPP-CPIM gateway.

8. IANA Considerations

8.1 Content-type Registration for "application/xmpp+xml"

To: ietf-types@iana.org

Subject: Registration of MIME media type application/xmpp+xml

MIME media type name: application

MIME subtype name: xmpp+xml

Required parameters: (none)

Optional parameters: charset Indicates the character encoding of the enclosed XML; the default encoding is UTF-8.

Encoding considerations: Contains XML, which can employ 8-bit characters, depending on the character encoding used.

Security considerations: Contains a message, presence information, or IQ (request-response) data in XMPP, which may be considered private. Appropriate precautions should be adopted to limit disclosure of this information.

Interoperability considerations: (none)

Specification: [RFCXXXX]

Applications which use this media type: XMPP-compliant instant messaging and presence systems.

Additional information: (none)

Person and email address to contact for further information: IETF, XMPP Working Group, <xmppwg@jabber.org>

Intended usage: COMMON

Author/Change controller: IETF, XMPP Working Group

8.2 XML Namespace Name for e2e Data in XMPP

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

URI: urn:ietf:params:xml:ns:xmpp-e2e

Specification: [RFCXXXX]

Description: This is the XML namespace name for signed and encrypted content in the Extensible Messaging and Presence Protocol as defined by [RFCXXXX].

Registrant Contact: IETF, XMPP Working Group, <xmppwg@jabber.org>

9. Security Considerations

This entire document discusses security. Detailed security considerations for instant messaging and presence protocols are given in RFC 2779 [3] (Sections 5.1 through 5.4), and for XMPP in particular are given in XMPP Core [1] (Sections 12.1 through 12.6).

The end-to-end security method defined here MAY result in exchanging secured instant messages and presence information through a gateway that implements the CPIM specifications. Such a gateway MUST be compliant with the minimum security requirements of the instant messaging and presence protocols with which it interfaces.

Normative References

[1] Saint-Andre, P. and J. Miller, "XMPP Core", draft-ietf-xmpp-core-17 (work in progress), August 2003.

[2] Saint-Andre, P. and J. Miller, "XMPP Instant Messaging", draft-ietf-xmpp-im-16 (work in progress), August 2003.

[3] Day, M., Aggarwal, S. and J. Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[4] Ramsdell, B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[5] Day, M., Rosenberg, J. and H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000, <http://www.ietf.org/rfc/rfc2778.txt>.

[6] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3369, August 2002.

[7] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[8] Atkins, D. and G. Klyne, "Common Presence and Instant Messaging: Message Format", draft-ietf-impp-cpim-msgfmt-08 (work in progress), January 2003.

[9] Fujimoto, S., Sugano, H., Klyne, G., Bateman, A., Carr, W. and J. Peterson, "Presence Information Data Format", draft-ietf-impp-cpim-pidf-08 (work in progress), May 2003.

[10] Crocker, D. and J. Peterson, "Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-im-03 (work in progress), May 2003.

[11] Crocker, D. and J. Peterson, "Common Profile for Presence (CPP)", draft-ietf-impp-pres-03 (work in progress), May 2003.

[12] Galvin, J., Murphy, S., Crocker, S. and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[13] Mealling, M., "The IANA XML Registry", draft-mealling-iana-xmlns-registry-05 (work in progress), June 2003.

[14] Ramsdell, B., "S/MIME Version 3 Certificate Handling", RFC

2632, June 1999.

[15] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

Informative References

[16] Saint-Andre, P. and T. Bamonti, "XMPP CPIM Mapping", draft-ietf-xmpp-cpim-02 (work in progress), August 2003.

Author's Address

Peter Saint-Andre
Jabber Software Foundation

EMail: stpeter@jabber.org

Appendix A. Schema for urn:ietf:params:xml:ns:xmpp-e2e

The following XML schema is descriptive, not normative.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:ietf:params:xml:ns:xmpp-e2e'
    xmlns='urn:ietf:params:xml:ns:xmpp-e2e'
    elementFormDefault='qualified'>

  <xs:element name='e2e' type='xs:string'/>

</xs:schema>
```

Appendix B. Revision History

Note to RFC Editor: please remove this entire appendix, and the corresponding entries in the table of contents, prior to publication.

B.1 Changes from draft-ietf-xmpp-e2e-04

- Added text about instant inbox addresses.

B.2 Changes from draft-ietf-xmpp-e2e-03

- Specified that S/MIME multipart objects are enclosed in a CDATA section.
- Changed "text/xml" to "text/plain" for message examples.
- Specified must-implement technologies, transfer encodings, certificate enrollment, certificate retrieval, and certificate names (including subjectAltName for JIDs).
- Specified requirements regarding attachment of signatures and inclusion of certificates.
- Fixed some small terminological errors.

B.3 Changes from draft-ietf-xmpp-e2e-02

- Completely revised to use formats defined in the CPIM specifications.

B.4 Changes from draft-ietf-xmpp-e2e-01

- Removed old Section 6 (Signalling Support via Presence) -- the ability to sign broadcasted presence made it redundant.
- Made small editorial changes to address RFC Editor requirements.

B.5 Changes from draft-ietf-xmpp-e2e-00

- Added support for all stanza types.
- Specified that the full stanza is encrypted.
- Added support for S/MIME in addition to OpenPGP.

- Specified that encrypted presence must be directed to a specific recipient.
- Specified order of encrypting and signing.
- Added support for signing broadcasted presence.
- Added IANA considerations.
- Changed namespace to 'urn:ietf:params:xml:ns:xmpp-e2e'.
- Added XML schema.

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.