Network Working Group
Internet-Draft
Expires: October 20, 2003

P. Saint-Andre
Jabber Software Foundation
J. Hildebrand
Jabber, Inc.
April 21, 2003

End-to-End Object Encryption in XMPP
draft-ietf-xmpp-e2e-02

Status of this Memo

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

This document describes an end-to-end signing and encryption method for use in the Extensible Messaging and Presence Protocol (XMPP) as defined by XMPP Core [1] and XMPP IM [2]. Object signing and encryption enable a sender to encrypt an XML stanza sent to a specific recipient, sign such a stanza, sign broadcasted presence, and signal support for the method defined herein. This document thereby helps the XMPP specifications meet the requirements defined in RFC 2779 [6].

1.1 Terminology

This document inherits the terminology defined in 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 [3].

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. Encryption must work with any stanza type (message, presence, or IQ).

2. The full XML stanza must be encrypted.

3. Encryption must be possible using either OpenPGP [4] or S/MIME [5].

4. It must be possible to sign encrypted content.

5. It must be possible to sign broadcasted presence.

6. Any namespaces used must conform to The IETF XML Registry [7].

3. Encrypting Stanzas

3.1 General Syntax

Any stanza MAY be encrypted. The full stanza MUST be inserted as a direct child of a <payload/> element scoped by the 'urn:ietf:params:xml:ns:xmpp-e2e#payload' namespace. The stanza data MUST be preceded by another direct child of the <payload/> element, namely an <id/> element. The CDATA of the <id/> element MUST be constructed according to the following algorithm:

1. concatenate the sender's full JID (user@host/resource) with the recipient's full JID

2. concatenate the resulting string with a full ISO-8601 UTC timestamp including year, month, day, hours, minutes, seconds in the following format: yyyy-mm-dd-Thh:mm:ssZ (the timestamp must be UTC, no offsets are allowed)

3. hash the resulting string according to the SHA1 algorithm

4. convert the hexidecimal SHA1 output to all lowercase.

Before encryption, the XML to be encrypted will thus be of the following form:

```
<payload xmlns='urn:ietf:params:xml:ns:xmpp-e2e#payload'>
  <id>someID</id>
  [stanza]
</payload>
```

The full <payload/> element (including all XML tag names and angle brackets) MUST then be encrypted using either OpenPGP or S/MIME. The output MUST be armored US-ASCII with any headers removed. The resulting cipher text MUST then be provided as the CDATA of the <stanza/> child of an <x/> element scoped by the 'urn:ietf:params:xml:ns:xmpp-e2e' namespace, with the value of the 'type' attribute set to either "openpgp" or "smime" depending on which method was used.

The format of the stanza that results from the foregoing procedure is as follows (no 'from' address is included in the stanza to be encrypted, since that is stamped by the sender's server):

```
<[stanza-name]
    to='recipient'
    type='[value if provided]'
    id='[value if provided]'
    xml:lang='[value if provided]'>
  <x type='[openpgp|smime]'
     xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
    <stanza>
      [encrypted content]
    </stanza>
  </x>
</[stanza-name]>
```

3.2 Encrypting Messages

Message stanzas may be encrypted using the syntax defined above.

Example: Sender generates encrypted message ('from' address is stamped by sender's server):

```
<message to='romeo@montague.net/orchard' type='chat'>
  <x type='openpgp' xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
    <stanza>
hQEOA+fczQLixGb6EAP/SmSRmrzpZQ9OPrjbS2HoZ4VkfNEodykB/TiDt86NdtPE
zmeLBduajZEQQhslUbBu8355fvy/ykDom1Xe/S1q56ZMEsSXkDO4x1xt/3OE/Hru
ovLXkTAVNX9pfTQb4rC2CC9G+X/ZsRiUf53ug/9PGBDMByiqWRWUBWipWqxoBbID
/2j83fQTGopp//tKijmhyMK7/xC73p/9TezvIz1ESqJY2NwSoRo0us6mKu4bBQ3G
EtOmMJZZUToNZwgDfLODzZHGOyiT4tdUL9eCln2a5FAgN75NnCUDHdRw0zpaCVIK
El389vMl8L0irlmxBMhVYLDyxAwsB8evXkAJeYu0mLuJ0sBZAbyfSlnGr8sAZ7c4
peSUpSBMhA4lAOnUASra2tYNsvOdfiFU2V7k1QEoR4c0HBB+ORX5HElPFdgzYM6Q
yhxSNWxTqBD1CfYSHM2KNzSJnEimSeL6/bhO32tAXIK+rigywLyCDAFEpYOjLXhp
9TA5pQw5ADMzmJnYlq3H5q4kn7s7RfzUuWflQjzhU4u2YFj3lJIRpO1szyXAACTG
hJbxpwL0I2Gz4YezWnzIKWU5xTna+V+0heP+lfUfmkP9CtTZZEmxEPKkWTnCt7Fk
wUlr9DeqqO5dGd+1KT94QY7clAnb7IRIgGP/ZeGQpn6A4XRvIDwe3/kMAdWLVSR7
aYHSCl6JG9ozHGlwIR3HF8K09je/oQwhXvnzimQ=
=zjBS
    </stanza>
  </x>
</message>
```

The decoded payload is:

```
<payload xmlns='urn:ietf:params:xml:ns:xmpp-e2e#payload'>
  <id>e0ffe42b28561960c6b12b944a092794b9683a38</id>
  <message
      to='romeo@montague.net/orchard'
      type='chat'>
    <subject>Imploring</subject>
    <body>O Romeo, Romeo! Wherefore art thou Romeo?</body>
  </message>
</payload>
```

3.3 Encrypting Presence

Presence stanzas may be encrypted using the syntax defined above. An encrypted presence stanza MUST be an instance of directed presence; i.e., the <presence/> element MUST possess a 'to' attribute specifying a specific intended recipient. Presence information that is intended for broadcasting to all subscribers MUST NOT be encrypted.

Example: Sender generates encrypted presence ('from' address is stamped by sender's server):

```
<presence to='romeo@montague.net/orchard'>
  <x type='openpgp' xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
    <stanza>
hQEOA+fczQLixGb6EAP/YOqZS+jgzDrXdqIyuDDJI2oxH2LXZf10LeR6EeBdGqX8
ewI8Nsf3CR4Mou58tRZ1QG5EOsCl6aylUxAiJuSe5f+Lv97dRWGQnrAQ4RNVpJ8O
jzPf+UQJ6mBZhgBgrtPB8XML7dORJqWBR69ralLcGhOtBr0CsNo7RyoZUWrfl74D
/0yJ7y3ZyHmA1gDRd9f7CZuMwdNF+xCfQtZjtAdc+t7HNsoJSNxGBeQdJbdpIaJo
jvHfiVG6jvrGDzWceyj4SnFkxOfxb+xu1x7mcmiXW0Jb58wsddttmhqBDdDd4B3H
QKnZCkyMPUcldzCBXUf4JPbC5EcUnNOmT6mth9+Qj0GJ0sAPAW2tZu5LOLVQU5Wo
zMJBZJOlaiyEv74YSYCjGNwKP9Yh+f+rBL1UkmnKqfiZVxSQo50ccPkJ45Syq85j
v8RSvYsU27bTQdCNL/ZS5aILQHryD2iXoLDk9XkzVDTBDNahOk1IWUaJwU5Qy1Lw
olEYwndAQi0ieXQklW+2HRmq5fZNslItCPJBGWmxAdGO6xyKbkbqCfq6ytw9kXjW
wAoBMgWZFfIbBh5EdBd7NO8u9bF3oDXxKO7c4dkg6WXUjJTZzEIWZCNaFa1PcW+3
/FoQ
=HT9r
    </stanza>
  </x>
</presence>
```

The decoded payload is:

```
<payload xmlns='urn:ietf:params:xml:ns:xmpp-e2e#payload'>
  <id>e0ffe42b28561960c6b12b944a092794b9683a38</id>
  <presence to='romeo@montague.net/orchard'>
    <show>away</show>
    <status>retired to the chamber</status>
  </presence>
</payload>
```

3.4 Encrypting IQs

IQ stanzas may be encrypted using the syntax defined above.

Example: Sender generates encrypted IQ ('from' address is stamped by sender's server):

```
<iq to='romeo@montague.net/orchard' type='get' id='eq7-2521'>
  <x type='openpgp' xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
    <stanza>
hQEOA+fczQLixGb6EAP/ejh9XMAbiFTA4WRIOyBXiiiAHtKCe/AKcn5I1M+HI/AR
8K3LdWbg4CzuBfDv/Sb9zesVXIZZEvHHQF6ihjxQpW0V0a1lvgDq49Dc0bR4uPsz
sFRr9auTnouZ5062ubwGk3Uic8CChC/JZxlfdRXO4ac3jS+uzafC0aJ9hkn0QkoD
/0b9PpTC3OYq5JoMpFSvBHeHOyixqKQh6xhBgJLzr2/6ZId/axOpgq7ru1GyYmHg
+dg/wuizJLgMaSSLwmEM58JiGKs44RHcQMUlnEruQvSbbCCNKIaLCMVQPLXS+oaD
Ly2ZG8BW+lb0j0d2E0dXbM30TvTfCW9w76xvOnX/BLRT0sA6AVmuJLz6+UN55roD
dE7HncBV0J/NmWksTHL/e4521O9aWSrqTYFsG6Fvvu7In5o0iKHIkLVzosW49zA9
McDna2krEtjWCsx5feHbxGrBWOpuPPHqD+uuSvD7f7RLWOKvW+Jz2/OXBOJUJ2+x
+xX9uaTdP08TlfBa4BrsX5mM+eFhkPC5oDg3O8Jy612A2Jf8IRQ4lYZDoz6SWoHl
scfHcSWjqont7hUTXtdTEnHcs9UkaxXlrbwLBaEfix0J7ALgjAESfEjG88eHm5oj
49I9rju8kw+HEsSl/moI+icDmuc0mN7bjOcKM3rIeU/roqWD0llWFIyWWrMNLg==
=6H0T
    </stanza>
  </x>
</iq>
```

The decoded payload is:

```
<payload xmlns='urn:ietf:params:xml:ns:xmpp-e2e#payload'>
  <id>e0ffe42b28561960c6b12b944a092794b9683a38</id>
  <iq to='romeo@montague.net/orchard' type='get' id='eq7-2521'>
    <query xmlns='jabber:iq:version'/>
  </iq>
</payload>
```

4. Signing Encrypted Content

OpenPGP and S/MIME both allow an entity to either encrypt then sign, or sign then encrypt. When signing first, the signatories are obscured by the encryption; when encrypting first, the signatories are exposed but the signatures can be verified without decrypting. Because in XMPP the signatories are exposed by the very act of exchanging a stanza (since the 'from' and 'to' addresses must be exposed for routing purposes), there would be no use in signing first and encrypting second. Therefore, if signing is desired, it SHOULD be performed after encrypting.

5. Signing Broadcasted Presence

An entity may want to sign presence information for broadcasting to all subscribers (i.e., the presence stanza is not directed to a particular recipient, but is sent to all other entites that that have subscribed to the sender's presence). Because encrypted presence MUST be directed to a particular recipient, signed presence for broadcasting MUST NOT be encrypted, only signed. However, there is little to no value in signing the entire stanza; therefore it is enough to sign only the user-provided CDATA of the <status/> element (note that this requires a signed presence broadcast to include some CDATA in the <status/> element). The process is as follows:

1. User provides CDATA for the <status/> element.

2. Client application signs CDATA using OpenPGP or S/MIME.

3. Client application inserts signed ASCII output as CDATA of the <signed/> child of an <x/> element that is scoped by the 'urn:ietf:params:xml:ns:xmpp-e2e' namespace and that includes a 'type' attribute whose value is either "openpgp" or "smime".

4. Client application adds <x/> element to remainder of presence stanza and sends to server with no 'to' attribute.

5. Server stamps 'from' address, enabling recipient to check bare JID (user@domain) of 'from' address against User-IDs defined in the sender's key or certificate.

Example: Sender generates signed presence:

```
<presence>
  <show>away</show>
  <status>retired to the chamber</status>
  <x type='openpgp' xmlns='urn:ietf:params:xml:ns:xmpp-e2e'>
    <signed>
iD8DBQE+kgpNEWF4x4jKHUYRAuthAJ9L1BjML9GIpagVGbEEJr0C7F3k9ACeJRL4
obxiSG72h3ggH0Xr3BmGyjE=
=T4rw
    </signed>
  </x>
</presence>
```

6. IANA Considerations

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>

7. Security Considerations

Replay attacks are made more difficult using this method because of the inclusion of a unique ID in the encrypted object. Key exchange may rely on the web of trust model used on the OpenPGP keys network. There is no method to check a fingerprint or ownership of a key other than checking the user IDs on a key. A key or certificate SHOULD have associated with it the Jabber ID of the sender. One of the User-IDs defined in a sender's key or certificate MUST be the bare JID (user@domain) of the 'from' address stamped by the sender's server on the XML stanzas that the sender generates; a client that receives signed or encypted stanzas from the sender MUST check the sender's bare JID against the User-IDs defined in the sender's key or certificate, and SHOULD discard the stanza or warn the recipient before presenting the stanza to the recipient if the bare JID does not match.

Normative References

[1] Saint-Andre, P. and J. Miller, "XMPP Core (draft-ietf-xmpp-core-10, work in progress)", April 2003.

[2] Saint-Andre, P. and J. Miller, "XMPP Instant Messaging (draft-ietf-xmpp-im-09, work in progress)", April 2003.

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

[4] Callas, J., Donnerhacke, L., Finney, H. and R. Thayer, "OpenPGP Message Format", RFC 2440, November 1998.

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

Informative References

[6] Day, M., Aggarwal, S., Mohr, G. and J. Vincent, "A Model for Presence and Instant Messaging", RFC 2779, February 2000, <http://www.ietf.org/rfc/rfc2779.txt>.

[7] Mealling, M., "The IANA XML Registry", draft-mealling-iana-xmlns-registry-04 (work in progress), June 2002.

Authors' Addresses

Peter Saint-Andre
Jabber Software Foundation

EMail: stpeter@jabber.org
URI: http://www.jabber.org/people/stpeter.php

Joe Hildebrand
Jabber, Inc.

EMail: jhildebrand@jabber.com
URI: http://www.jabber.org/people/hildjj.php

Appendix A. XML Schemas

The following XML schemas are descriptive, not normative.

A.1 urn:ietf:params:xml:ns:xmpp-e2e

```
<?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='x'>
    <xs:complexType>
      <xs:choice>
        <xs:element ref='signed' minOccurs='0' maxOccurs='1'/>
        <xs:element ref='stanza' minOccurs='0' maxOccurs='1'/>
      </xs:choice>
      <xs:attribute name='type' use='required'/>
        <xs:simpleType>
          <xs:restriction base='xs:NCName'>
            <xs:enumeration value='openpgp'/>
            <xs:enumeration value='smime'/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name='signed' type='xs:string'/>
  <xs:element name='stanza' type='xs:string'/>

</xs:schema>
```

A.2 urn:ietf:params:xml:ns:xmpp-e2e#payload

```
<?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#payload'
    xmlns='urn:ietf:params:xml:ns:xmpp-e2e#payload'
    elementFormDefault='qualified'>

  <xs:element name='payload'>
    <xs:complexType>
      <xs:element ref='id' maxOccurs='1'/>
      <xs:any namespace='jabber:client' maxOccurs='1'/>
    </xs:complexType>
  </xs:element>

  <xs:element name='id' 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-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.2 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.

Full Copyright Statement

Acknowledgement

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