Network Working Group	M. Miller	
Internet-Draft	P. Saint-Andre	
Obsoletes: 3923 (if approved)	Cisco Systems, Inc.	
Intended status: Standards Track	June 29, 2010	
Expires: December 31, 2010		

TOC

End-to-End Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)
draft-miller-3923bis-02

Abstract

This document defines a method of end-to-end object encryption for the Extensible Messaging and Presence Protocol (XMPP). The protocol defined herein is a simplified version of the protocol defined in RFC 3923.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as “work in progress.”
This Internet-Draft will expire on December 31, 2010.

Copyright Notice

Table of Contents

1. Introduction

TOC

End-to-end encryption of traffic sent over the Extensible Messaging and Presence Protocol [XMPP-CORE] (Saint-Andre, P., “Extensible Messaging and Presence Protocol (XMPP): Core,” May 2010.) is a desirable goal. Requirements and a threat analysis for XMPP encryption are provided in [E2E-REQ] (Saint-Andre, P., “Requirements for End-to-End Encryption in the Extensible Messaging and Presence Protocol (XMPP),” March 2010.). Many possible approaches to meet those (or similar) requirements have been proposed over the years, including methods based on PGP, S/MIME, SIGMA, and TLS.

The S/MIME approach defined in [RFC3923] (Saint-Andre, P., “End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP),” October 2004.) has never been implemented in XMPP clients to the best of our knowledge, but has some attractive features, especially the ability to store-and-forward an encrypted message at a user's server if the user is not online when the message is received (in the XMPP community this is called "offline storage" and the message is referred to as an "offline message"). The authors surmise that RFC 3923 has not been implemented mainly because it adds several new dependencies to XMPP clients, especially MIME (along with the CPIM and MSGFMT media types). Therefore this document explores the possibility of an approach that is similar to but simpler than RFC 3923, while retaining the same basic object encryption model.

2. Terminology

TOC

This document inherits terminology defined in [XMPP-CORE] (Saint-Andre, P., “Extensible Messaging and Presence Protocol (XMPP): Core,” May 2010.).
Security-related terms are to be understood in the sense defined in [SECTERMS] (Shirey, R., “Internet Security Glossary, Version 2,” August 2007.).
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 [KEYWORDS] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

3. Securing XMPP Stanzas

TOC

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

1. Constructs a cleartext version of the stanza, S.

2. Generates a session key R appropriate for the intended block cipher (e.g. AES-SHA-256).

3. Notes the current UTC date and time N when this stanza is constructed, formatted as described under Section 6 (Inclusion and Checking of Timestamps).

4. Converts the stanza to a UTF-8 encoded string, optionally removing line breaks and other insignificant whitespace between elements and attributes, i.e., S' = UTF8-encode(S). We call S' a "stanza-string" because for purposes of encryption and decryption it is treated not as XML but as an opaque string (this avoids the need for complex canonicalization of the XML input).

5. Constructs a plaintext envelope (E) <plain/> as follows:

   *The attribute 'timestamp' set to the UTC date and time value N

   *The XML character data set to the base64-encoded form of S' (where the encoding adheres to the definition in Section 4 of [BASE64] (Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," October 2006.) and where the padding bits are set to zero). This encoding is necessary to preserve a canonicalized form of S'.

6. Converts the envelope (E) to a UTF-8 encoded string, optionally removing line breaks and other insignificant whitespace between elements and attributes, i.e., E' = UTF8-encode(E).

7. Encrypts the UTF8-encoded enveloped (E') using the intended block cipher, i.e. T = block-encrypt(R, E').

8. Generates a message authentication code (MAC) with a cryptographic hashing algorithm (e.g. HMACSHA256) using the encrypted data T as the salt and the session block cipher key R as the message, i.e., M = mac-hash(T, R)

9. Encrypts the session key (R) using the recipient's public key to produce encrypted data K. (Known issue: This step is under-specified and will be expanded in a later version of this document.)

10. Constructs an <e2e/> element qualified by the "urn:ietf:params:xml:ns:xmpp-objenc:0" namespace as follows:

    *The child element <key/> (implicitly qualified by the "urn:ietf:params:xml:ns:xmpp-objenc:0" namespace) as follows:

      -The attribute 'cipher-algo' set to the asynchronous encryption scheme used in step 9;

      -The XML character data set to the base64-encoded form of K.

*The child element <data/> qualified by the
"urn:ietf:params:xml:ns:xmpp-objenc:0" namespace as follows:

-The attribute 'mac-algo' set to the cryptographic hashing algorithm used to generate M in step 8;

-The attribute 'mac-hash' set to the base64-encoded result of the MAC, M;

-The attribute 'cipher-algo' set to the block encryption scheme used to generate the encrypted data T in step 7;

-The XML character data as the base64-encoded form of T.

11. Sends the <e2e/> element as the payload of a stanza that SHOULD match the stanza from step 1 in kind (e.g., <message/>), type (e.g., "chat"), and addressing (e.g. to="romeo@montague.net" from="juliet@capulet.net/balcony"). If the original stanza (S) has a value for the "id" attribute, this stanza MUST NOT use the same value for its "id" attribute.

3.1. Example of Securing Messages

TOC

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

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

The sender then performs the steps 1 through 5 from above to generate:

```
<plain  xmlns="urn:ietf:params:xml:ns:xmpp-objenc:0"
        timestamp="2010-06-29T02:15:21.012Z">
  PG1lc3NhZ2UgeG1sbnM9ImphYmJlcjpjbGllbnQiIGZyb209Imp1bGlldEBjYXB
  1bGV0Lm5ldC9iYWxjb255IiB0bz0icm9tZW9AbW9udGVndWUubmV0IiB0eXBlPS
  JjaGF0Ij48dGhyZWFkPmM2MzczODI0LWEzMDctNDBkZC04ZmUwLWJhZDZlNzI5O
  WFkMDwvdGhyZWFkPjxib2R5PldoZXJlZm9yZSBhcnQgdGhvdSwgUm9tZW8/PC9i
  b2R5PjwvbWVzc2FnZT4=
</plain>
```

Then performs steps 6 through 10, and sends the following:

```
<message  xmlns='jabber:client'
          from='juliet@capulet.net/balcony'
          id='6410ed123'
          to='romeo@montague.net'
          type='chat'>
  <e2e   xmlns="urn:ietf:params:xml:ns:xmpp-objenc:0">
    <key    cipher-algo="RSAES-OAEP-SHA-256-MFG1">
      OPfr4zudqiEeLcOQazZJIB6B9gx3zrVbyHKTU8a/aDb0wiZevztxxCi8hto0+Qw
      Foyhcupj547WbFZJNlB2dsAPhlJzeH9SuGLJShjhbkOyKjmqZLLCZr3OQtJjcTU
      sAVj7IZZsOOPDmwsb4Dxv5sz+icsDpi5l+5APfthDaoHbcrvz2pA1CJ5IFQoob4
      a0i0WevcAFyB+vWXsRqQCxjn5sHdb6G4vjQ/m1lzTWahzKvi56pNUm7ll18oI8L
      mPi1VWUEqH3aayGLVlJ9fhBDSSpW4jTQ/ts1nzPJwVlKdTqdgNBusFEhrRMhJD5
      1JdLOhxx+Ov2Xbs22++XQ1tS8/A==
    </key>
    <data   cipher-algo="AES-256-CBC-PKCS5-WITH-IV"
            mac-algo="HMACSHA256"
            mac-data="HSGmwUFd4sESB+O12S32xsXVvMnO4gjRPaQITIrjWbs=">
      a8zpjgRcO1VHZ9CoqU19/jB7nn58Gzu5/sQm8YQe4F9zz+YKUfqTS9LaHcqdAwa
      z8BG1a24Z72VYb5Ptjh7nQ19f5QQdA/P4lZ3oqeTJTsA4DkhvJaSUhrjYib/NOk
      3lkMoatR/OSbfvhPdqXQ/dutLuRFjkilXGVwNWkgLm3iSnKUiYSdUzWvj88RgR3
      ldVHFeyrdgufU9qu/FyO6MZXjfEtD80O+3ZBbESqllzmYFXnfkzBrhfi14iCba6
      /b5Io5zhFUyWaq5e6qq2z72a+1bjeWkG8F9XBiMkyaxkB64wAS0o6aDpWdir5Oi
      +Rnms4LV/wxL4Is/oe8Fo9xR3UmrdlAiaehdGBh+EnJGqprKa9eccOKqSu7/lJQ
      ObAdJGEOeAVs8JEkQkxw+qR8edkEDuv6ZXN7JCWQx9LNaiiwsfAzApJJbqfrtDx
      koQ3JaBbxQ+8FE3TM0E4Tbr9V8NDZC8abgBramlpUBfgknJvLYMTzx1lnsiCUxo
      6ezC0xqV
    </data>
  </e2e>
</message>
```

3.2. Example of Securing IQs

TOC

The sender begins with the cleartext version of the <iq/> stanza "S":

```
<iq     xmlns="jabber:client"
        from="juliet@capulet.net/crypt"
        id="a543bc3ee"
        to="romeo@montegue.net/crypt"
        type="result">
  <mood xmlns="http://jabber.org/protocol/mood">
    <dejected />
    <text>
      Romeo, what's here? Poison? Drunk all, and
      left no friendly drop to help me after?
    </text>
  </mood>
</iq>
```

The sender then performs the steps 1 through 5 from above to generate:

```
<plain  xmlns="urn:ietf:params:xml:ns:xmpp-objenc:0"
        timestamp="2010-06-29T02:15:21.012Z">
  PGlxIHhtbG5zPSJqYWJiZXI6aXEiIGZyb209Imp1bGlldEBjYXB1bGV0Lm5ldC9
  jcnlwdCIgaWQ9ImE1NDNiYzNlZSIgdG89InJvbWVvQG1vbnRlZ3VlLm5ldC9jcn
  lwdCIgdHlwZT0icmVzdWx0Ij48bW9vZCB4bWxucz0iaHR0cDovL2phYmJlci5vc
  mcvcHJvdG9jb2wvbW9vZCI+PGRlamVjdGVkIC8+PHRleHQ+Um9tZW8sIHdoYXQn
  cyBoZXJlPyBQb2lzb24/IERydW5rIGFsbCwgYW5kIGxlZnQgbm8gZnJlbmRseSB
  kcm9wIHRvIGhlbHAgbWUgYWZ0ZXI/PC90ZXh0PjwvbW9vZD48L2lxPg==
</plain>
```

Then performs steps 6 through 10, and sends the following:

```
<iq     xmlns="jabber:client"
        type="result"
        to="romeo@montegue.net/crypt"
        id="42ca3de0345"
        from="juliet@capulet.net/crypt">
  <e2e xmlns="urn:ietf:params:xml:ns:xmpp-objenc:0">
    <key    cipher-algo="RSAES-OAEP-SHA-256-MFG1">
      hOU+BRkEcCY0+eKTX9hzCbP30Ij0q5zZ9buFgkOWu4LsVkI92OiH65SvYL/XCB6
      12sb9fhjkiAIeR0AySGiid+AeS7KZDzpcZ+ORg8j9CkEX/LeTYszBfZFiHzDFkh
      qtwu3s7QMAR0Bzxj9NVE7W8fSdleusvyOOP5c0scrpRkXDMVO2Z3/rTjC0xInx3
      XQUP+RlqFE7g1HCr01BjoPjI4p3N+fONVv0U9mwtt1I5tJ4EXgTofUM0GMNGX1i
      NoNNjPDb9XsihpLvDIjMblXVHvYAIyPwCs2ZdDv7L5kmZ6U+35b7Qx8TdWUN2I4
      5fBbxczvkFN6+cx2h5uapOTxBkw==
    </key>
    <data   cipher-algo="AES-256-CBC-PKCS5-WITH-IV"
            mac-algo="HMACSHA256"
            mac-data="iKuTGRZNHe3PbZNdfxkFzwClXLMQlhx0Y8BuYawbaho=">
      ksCAkoJeoymtf3ygzBJkrJYQV+g04CkAs5oSmej60GU89mRN3rKSX5FVfWo558W
      Bcn8mVUxFxWhSdNBrsW5GQS1EyygDT+yfJe6OqzLTCqZn4iqaCyIPWM7XB/PolA
      fvELw7y3hf8JrEAM4JXIfxrcOYDqewr7zmamwuuos4B6qzgiNN9ZW2AfTyKL3+l
      twcmFvF/nWF1YN8CquGmBm83WFn7Ik9R+Nqq54+QNCABjSFPT25ZYquEhxk/RIS
      CDAIXFOaFBozGjC20M3UDqnuwLsUF+P4ucjybysxhHQlqLOffX0Vhb1YesWaZac
      pvsj8Ovfpv+ESrWGptXr+8GMK1og69GHRrd2k2TonPFp1KwS5MkbEpP2tS7R+nT
      b9oGFojr6waNKhhhVmP/9FWRMi7C2KfLCHggAatLWDjBG8k7yd5DWdSqY7LwkwB
      hT6+iErRfhdvk1EVxn2TVqjfhsFh33XDqkRT4BhPJUjJPkwLZkQ03PVgHKluMsE
      JoUBS0OxD7gE5q808hy3qA+r5PDowy6nQ9zbUaCu4JbvKV2moql7fgHUy8MZLIe
      DFVJ5A5z8Te6K4pFaQGAzxEOouS2A+BmvPAFczFeL+QGy58RSNMiXJ9ZMpb+N2C
      1iDzPD8OL
    </data>
  </e2e>
</iq>
```

---

4. Interaction with Stanza Semantics

TOC

The following limitations and caveats apply:

*Undirected <presence/> stanzas MUST NOT be encrypted. Such stanzas are delivered to anyone the sender has authorized, and therefore it is highly unlikely that the sender can find an appropriate certificate.

*Stanzas directed to multiplexing services (e.g. multi-user chat) SHOULD NOT be encrypted, unless the sender has established an acceptable trust relationship with the multiplexing service.

5. Handling of Inbound Stanzas

TOC

Several scenarios are possible when an entity receives an encrypted stanza:

- The receiving application does not understand the protocol.
- The receiving application understands the protocol and is able to decrypt the payload.
- The receiving application understands the protocol and is able to decrypt the payload, but the timestamps fail the checks specified under Checking of Timestamps (Inclusion and Checking of Timestamps).
- The receiving application understands the protocol but is unable to decrypt the payload.

In Case #1, the receiving application MUST do one and only one of the following: (1) ignore the <e2e/> extension, (2) ignore the entire stanza, or (3) return a <service-unavailable/> error to the sender, as described in [XMPP-CORE] (Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," May 2010.).
In Case #2, the receiving application MUST NOT return a stanza error to the sender, since this is the success case.
In Case #3, the receiving application MAY return a <not-acceptable/> error to the sender (as described in [XMPP-CORE] (Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," May 2010.)), optionally supplemented by an application-specific error condition element of <bad-timestamp/> (previously defined in [RFC3923] (Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)," October 2004.)):

```
<message from='romeo@example.net/orchard'
         id='6410ed123'
         to='juliet@capulet.net/balcony'
         type='error'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-objenc:0'>
    XML-character-data-here
  </e2e>
  <error type='modify'>
    <not-acceptable xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <bad-timestamp xmlns='urn:ietf:params:xml:xmpp-e2e'/>
  </error>
</message>
```

In Case #4, the receiving application SHOULD return a <bad-request/> error to the sender (as described in [XMPP-CORE] (Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," May 2010.)), optionally supplemented by an application-specific error condition element of <decryption-failed/> (previously defined in [RFC3923] (Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)," October 2004.)):

```
<message from='romeo@example.net/orchard'
         id='6410ed123'
         to='juliet@capulet.net/balcony'
         type='error'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-objenc:0'>
    XML-character-data-here
  </e2e>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <decryption-failed xmlns='urn:ietf:params:xml:xmpp-e2e'/>
  </error>
</message>
```

In addition to returning an error in Case #4, the receiving application SHOULD NOT present the stanza to the intended recipient (human or application) and SHOULD provide some explicit alternate processing of the stanza (which MAY be to display a message informing the recipient that it has received a stanza that cannot be decrypted).

---

6. Inclusion and Checking of Timestamps

TOC

Timestamps are included to help prevent replay attacks. All timestamps MUST conform to [DATETIME] (Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps," July 2002.) and be presented as UTC with no offset, and SHOULD include the seconds and fractions of a second to

three digits. 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.

- It SHOULD verify that the timestamp received is greater than any timestamp received in the last 10 minutes which passed the previous check.

- If any of the foregoing checks fails, the timestamp SHOULD be presented to the receiving entity (human or application) marked as "old timestamp", "future timestamp", or "decreasing timestamp", and the receiving entity MAY return a stanza error to the sender.

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 [OFFLINE] (Saint-Andre, P., "Best Practices for Handling Offline Messages," January 2006.), when sending an offline message to the recipient, the server SHOULD include delayed delivery data as specified in [DELAY] (Saint-Andre, P., "Delayed Delivery," September 2009.) 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.

---

7. Mandatory-to-Implement Cryptographic Algorithms

TOC

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

- The RSA (PKCS #1 v2.1) key transport, as specified in [X509-ALGO] (Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1," February 2003.).

*The AES-128 encryption algorithm in CBC mode, as specified in [CMS-AES] (Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)," July 2003.).

*The HMACSHA256 hashing algorithm, as specified in [HMAC] (Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)," July 2006.).

---

8. Certificates

TOC

To participate in end-to-end encryption using the methods defined in this document, a client needs to possess an X.509 certificate [PKIX] (Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," May 2008.). It is expected that many clients will generate their 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 [XMPP-CORE] (Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," May 2010.).

---

9. Security Considerations

TOC

The recipient's server might store any <message/> stanzas received until the recipient is next available; this duration could be anywhere from a few minutes to several months.

---

10. IANA Considerations

TOC

TOC

10.1. XML Namespace Name for e2e Data in XMPP

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

> **URI:** urn:ietf:params:xml:ns:xmpp-objenc:0
>
> **Specification:** RFC XXXX
>
> **Description:** This is an XML namespace name of signed and encrypted content for the Extensible Messaging and Presence Protocol as defined by RFC XXXX.
>
> **Registrant Contact:** IESG, <iesg@ietf.org>

---

11. References

TOC

11.1. Normative References

TOC

[BASE64]	Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," RFC 4648, October 2006 (TXT).
[CMS-AES]	Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)," RFC 3565, July 2003 (TXT).
[DATETIME]	Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps," RFC 3339, July 2002 (TXT).
[DELAY]	Saint-Andre, P., "Delayed Delivery," XSF XEP 0203, September 2009.
[E2E-REQ]	Saint-Andre, P., "Requirements for End-to-End Encryption in the Extensible Messaging and Presence Protocol (XMPP)," draft-saintandre-xmpp-e2e-requirements-01 (work in progress), March 2010 (TXT).
[KEYWORDS]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.
[HMAC]	Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)," RFC 4634, July 2006 (TXT).
[PKIX]	Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, May 2008 (TXT).
[SECTERMS]	Shirey, R., "Internet Security Glossary, Version 2," RFC 4949, August 2007 (TXT).

[X509-ALGO]	Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1," RFC 3447, February 2003 (TXT).
[XMPP-CORE]	Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core," draft-ietf-xmpp-3920bis-08 (work in progress), May 2010 (TXT).

11.2. Informative References

TOC

[OFFLINE]	Saint-Andre, P., "Best Practices for Handling Offline Messages," XSF XEP 0160, January 2006.
[RFC3923]	Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)," RFC 3923, October 2004 (TXT).

Appendix A. Schema for urn:ietf:params:xml:ns:xmpp-objenc:0

TOC

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-objenc:0'
    xmlns='urn:ietf:params:xml:ns:xmpp-objenc:0'
    elementFormDefault='qualified'>

  <xs:element name='e2e'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='key' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='data' minOccurs='1' maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='key'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
          <xs:attribute name='cipher-algo'
                        type='xs:string'/>
        </xs:extension>
      </xs:simpleType>
    </xs:complexType>
  </xs:element>

  <xs:element name='data'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
          <xs:attribute name='cipher-algo'
                        type='xs:string'/>
          <xs:attribute name='mac-algo'
                        type='xs:string'/>
          <xs:attribute name='mac-hash'
                        type='xs:string'/>
        </xs:extension>
      </xs:simpleType>
    </xs:complexType>
  </xs:element>

  <xs:element name='plain'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
          <xs:attribute name='timestamp'
```

```
                        type='xs:string'/>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

    <xs:element name='decryption-failed' type='empty'/>
    <xs:element name='bad-timestamp' type='empty'/>

    <xs:simpleType name='empty'>
      <xs:restriction base='xs:string'>
        <xs:enumeration value=''/>
      </xs:restriction>
    </xs:simpleType>

  </xs:schema>
```

Authors' Addresses

TOC

	Matthew Miller
	Cisco Systems, Inc.
	1899 Wyknoop Street, Suite 600
	Denver, CO 80202
	USA
Phone:	+1-303-308-3204
Email:	mamille2@cisco.com
	Peter Saint-Andre
	Cisco Systems, Inc.
	1899 Wyknoop Street, Suite 600
	Denver, CO 80202
	USA
Phone:	+1-303-308-3282
Email:	psaintan@cisco.com