XMPP
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2013

M. Miller
Cisco Systems, Inc.

February 12, 2013

End-to-End Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)
draft-miller-xmpp-e2e-04

Abstract

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

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 August 16, 2013.

Table of Contents

1. Introduction

End-to-end encryption of traffic sent over the Extensible Messaging and Presence Protocol [RFC6120] is a desirable goal. Requirements and a threat analysis for XMPP encryption are provided in [E2E-REQ]. 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.

Most proposals have not been able to support multiple end-points for a given recipient. As more devices support XMPP, it becomes more desirable to allow an entity to communicate with another in a more secure manner, regardless of the number of agents the entity is employing. This document specifies an approach for encrypting

communications between two entities which each might have multiple end-points.

2. Terminology

This document inherits XMPP-related terminology from [RFC6120], JSON Web Algorithms (JWA)-related terminology from [JOSE-JWA], JSON Web Encryption (JWE)-related terminology from [JOSE-JWE], and JSON Web Key (JWK)-related terminology from [JOSE-JWK]. Security-related terms are to be understood in the sense defined in [RFC4949].

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 [RFC2119].

3. Determining Support

If an agent supports end-to-end object encryption, it MUST advertize that fact in its responses to [XEP-0030] information ("disco#info") requests by returning a feature of "urn:ietf:params:xml:ns:xmpp-e2e:5".

```
<iq xmlns='jabber:client'
    id='disco1'
    to='romeo@montegue.lit/garden'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'/>
    ...
  </query>
</iq>
```

To help facilitate discovery, an agent SHOULD also include [XEP-0115] information in any directed or broadcast presence updates.

4. Encrypting XMPP Stanzas

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/>).

4.1. Prerequisites

First, the sending agent prepares and retains the following:

- The JID of the sender (i.e. its own JID). This SHOULD be the bare JID (localpart@domainpart).

- The JID of the recipient. This SHOULD be the bare JID (localpart@domainpart).

- A Session Master Key (SMK). The SMK MUST have a length at least equal to that required by the key wrapping algorithm in use and MUST be generated randomly. See [RFC4086] for considerations on generating random values.

- A SMK identifier (SID). The SID MUST be unique for a given (sender, recipient, SMK) tuple, and MUST NOT be derived from SMK itself.

4.2. Process

For a given plaintext stanza (S), the sending agent performs the following:

1. Ensures the plaintext stanza is fully qualified, including the proper namespace declarations (e.g., contains the attribute 'xmlns' set to the value "jabber:client" for 'jabber:client' stanzas defined in [RFC6120]).

2. Notes the current UTC date and time (N) when this stanza is constructed, formatted as described under Section 7.

3. Constructs a forwarding envelope (M) using a <forwarded/> element qualified by the "urn:xmpp:forward:0" namespace (as defined in [XEP-0297]) as follows:

    * The child element <delay/> qualified by the "urn:xmpp:delay" namespace (as defined in [XEP-0203]) with the attribute 'stamp' set to the UTC date and time value N

    * The plaintext stanza S

4. Converts the forwarding envelope (M) to a UTF-8 encoded string (M'), optionallly removing line breaks and other insignificant whitespace between elements and attributes, i.e. M' = UTF8-encode(M). We call M' 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. Generates a Content Master Key (CMK). The CMK MUST have a length at least equal to that required by the content encryption algorithm in use and MUST be generated randomly. See [RFC4086] for considerations on generating random values.

6. Generates any additional unprotected block cipher factors (IV); e.g., initialization vector/nonce. A sending agent MUST ensure that no two sets of factors are used with the same CMK, and SHOULD NOT reuse such factors for other stanzas.

7. Performs the message encryption steps from [JOSE-JWE] to generate the JWE Header (H), JWE Encrypted Key (E), JWE Ciphertext (C), and JWE Integrity Value (I); using the following inputs:

   * The 'alg' property is set to an appropriate key wrapping algorithm (e.g., "A256KW" or "A128KW"); recipients use 'keyreq' in Section 6 to obtain the SMK.

   * The 'enc' property is set to the intended content encryption algorithm.

   * SMK as the key for CMK Encryption.

   * CMK as the JWE Content Master Key.

   * IV as the JWE Initialization Vector.

   * M' as the plaintext content to encrypt.

8. Constructs an <e2e/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace as follows:

   * The attribute 'id' set to the identifier value SID.

   * The child element <header/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as H, encoded base64url as per [RFC4648].

   * The child element <cmk/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character as E, encoded base64url as per [RFC4648].

   * The child element <iv/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character as IV, encoded base64url as per [RFC4648].

   * The child element <data/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as C, encoded base64url as per [RFC4648].

   * The child element <mac/> qualified by the "urn"ietf:params:xml:ns:xmpp-e2e:4" namespace and with XML character data as I, encoded base64url as per [RFC4648].

9. 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.

4.3. Example - Securing a Message

NOTE: unless otherwise indicated, all line breaks are included for readability.

The sending agent begins with the plaintext version of the <message/> stanza 'S':

```
<message xmlns='jabber:client'
         from='juliet@capulet.lit/balcony'
         to='romeo@montegue.lit'
         type='chat'>
  <thread>35740be5-b5a4-4c4e-962a-a03b14ed92f4</thread>
  <body>
    But to be frank, and give it thee again.
    And yet I wish but for the thing I have.
    My bounty is as boundless as the sea,
    My love as deep; the more I give to thee,
    The more I have, for both are infinite.
  </body>
</message>
```

and the following prerequisites:

- Sender JID as "juliet@capulet.lit/balcony"

- Recipient JID as "romeo@montegue.lit"

- Session Master Key 'SMK' as (base64 encoded)
  "xWtdjhYsH4Va_9SfYSefsJfZu03m5RrbXo_UavxxeU8"

- SMK identifier SID as "835c92a8-94cd-4e96-b3f3-b2e75a438f92"

The sending agent performs steps 1, 2, and 3 to generate the envelope:

```
<forwarded xmlns='urn:xmpp:forward:0'>
  <delay xmlns='urn:xmpp:delay'
         stamp='1492-05-12T20:07:37.012Z'/>
  <message xmlns='jabber:client'
           from='juliet@capulet.lit/balcony'
           to='romeo@montegue.lit'
           type='chat'>
    <thread>35740be5-b5a4-4c4e-962a-a03b14ed92f4</thread>
    <body>
      But to be frank, and give it thee again.
      And yet I wish but for the thing I have.
      My bounty is as boundless as the sea,
      My love as deep; the more I give to thee,
      The more I have, for both are infinite.
    </body>
```

```
  </message>
</forwarded>
```

Then the sending agent performs steps 4 through 7 (with Content Master Key as "upIjc_ePSomSETgi0DEnXsoT8ZEGf0QxsSHr_eDZRnlkJAJBFyenb6tm1WDAoqFD7-BHBtWqO5hOJlj2oxlDwQ", base64url encoded) to generate the [JOSE-JWE] outputs:

JWE Header

```
{
  "alg":"A256KW",
  "enc":"A256CBC+HS512",
  "kid":"835c92a8-94cd-4e96-b3f3-b2e75a438f92"
}
```

JWE Encrypted Key

```
4ui5xwE1gEYjuptNgSIaMFlwWrAOxMqBkaplTxeJ6b2iT8kQP2HHy5PYpqqmDxl
QgT5I5rO9mgAD7AUJ9Lx35fGdi5CMiRww
```

JWE Initialization Vector

```
B7waCj2vF_sLaJfe-1GHrA
```

JWE Ciphertext

```
UYbe-ziNGBL74581rynr9MWu0Ble_6M5LFCH9xOYXgALtlDih28Ilmf-Rs68uaZ
sOND-7Ii9zK4H4XBwJjxaUlDGChZPWdwotRZdQKt9ZLpiQmjkzrQgKVQqyexP6m
qhfWRHutEKgs6vR32O2P98J-4LAWoUza5qYCZHP5NCogLUBVKi-v-vGpHDKBG_S
w3ejHSXuZOEZtyXShL2d6EXOhEzft8ViaTUKhiBCLz1q39hI5TsPdS7NPHGQDUX
Db_gSw8yVCiJgxcSbfWJKj9v_zIZgxawZby6-qif7vTIizluirnSTRO-5-2xM_n
sJEpG7ZOqofzp_WKLPkOQfa8roYGp61R5BK2M3q9LKM6y1XlMrtYFyPWH70bVPC
S_kOMrrn_48G7zmPEl-2SZWrBj4llu0oPzO2EU4uh3ipb_xUwkPPQfTkwxEdcdl
Cbi4FFIQtw81_7bPwZ3m799O_-aPspkk4uFn_cKayeN3XKf8T-i9pYPWYEOugGq
GU3H0I-jfwvqt2K6GGctoXWD6-d56WFlLhv4v6qGPT5C30vO-xM22BU9nwc-rff
4Q7cFBBM_7ciZrrTQf_PBjBhWS_pTYsmIUL-h7dwhcgQ1LEdgpqAWbZ23aMDWx-
RSQSkRY6OlPYKkbrXUbXHWxlgb5B76eA
```

JWE Integrity Value

G5csTEYKIXipYM1Ey4_4JSUeHpgpd8lMvYxTHwPvSd7w916w0Q8VQekY1tz8VnA
DJ751V6YiJ295_3jQUphxmQ

Then the sending agent performs steps 8 and 9, and sends the following:

```
<message xmlns='jabber:client'
         from='juliet@capulet.lit/balcony'
         id='fJZd9WFIIwNjFctT'
         to='romeo@montegue.lit'
         type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
       id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>
      eyJhbGciOiJBMjU2S1ciLCJlbmMiOiJBMjU2Q0JDK0hTNTEyIiwia2lkI
      joiODM1YzkyYTgtOTRjZC00ZTk2LWIzZjMtYjJlNzVhNDM4ZjkyIn0
    </header>
    <cmk>
      4ui5xwE1gEYjuptNgSIaMFlwWrAOxMqBkaplTxeJ6b2iT8kQP2HHy5PYp
      qqmDxlQgT5I5rO9mgAD7AUJ9Lx35fGdi5CMiRww
    </cmk>
    <iv>
      B7waCj2vF_sLaJfe-1GHrA
    </iv>
    <data>
      UYbe-ziNGBL74581rynr9MWu0Ble_6M5LFCH9xOYXgALtlDih28Ilmf-R
      s68uaZsOND-7Ii9zK4H4XBwJjxaUlDGChZPWdwotRZdQKt9ZLpiQmjkzr
      QgKVQqyexP6mqhfWRHutEKgs6vR32O2P98J-4LAWoUza5qYCZHP5NCogL
      UBVKi-v-vGpHDKBG_Sw3ejHSXuZOEZtyXShL2d6EXOhEzft8ViaTUKhiB
      CLz1q39hI5TsPdS7NPHGQDUXDb_gSw8yVCiJgxcSbfWJKj9v_zIZgxawZ
      by6-qif7vTIizluirnSTRO-5-2xM_nsJEpG7ZOqofzp_WKLPkOQfa8roY
      Gp61R5BK2M3q9LKM6y1XlMrtYFyPWH70bVPCS_kOMrrn_48G7zmPEl-2S
      ZWrBj4llu0oPzO2EU4uh3ipb_xUwkPPQfTkwxEdcdlCbi4FFIQtw81_7b
      PwZ3m799O_-aPspkk4uFn_cKayeN3XKf8T-i9pYPWYEOugGqGU3H0I-jf
      wvqt2K6GGctoXWD6-d56WFlLhv4v6qGPT5C30vO-xM22BU9nwc-rff4Q7
      cFBBM_7ciZrrTQf_PBjBhWS_pTYsmIUL-h7dwhcgQ1LEdgpqAWbZ23aMD
      Wx-RSQSkRY6OlPYKkbrXUbXHWxlgb5B76eA
    </data>
    <mac>
      G5csTEYKIXipYM1Ey4_4JSUeHpgpd8lMvYxTHwPvSd7w916w0Q8VQekY1
      tz8VnADJ751V6YiJ295_3jQUphxmQ
    </mac>
  </e2e>
</message>
```

5. Decrypting XMPP Stanzas

5.1. Protocol Not Understood

If the receiving agent does not understand the protocol, it 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 [RFC6120].

NOTE: If the inbound stanza is an <iq/>, the receiving agent MUST return an error to the sending agent, to comply with the exchanging of IQ stanzas in [RFC6121].

5.2. Process

Upon receipt of an encrypted stanza, the receiving agent performs the following:

1. Determines if a valid SMK is available, associated with the SID specified by the 'id' attribute value of the <e2e/> element and the sending agent JID specified by the 'from' attribute of the wrapping stanza. If the receiving agent does not already have the CMK, it requests it according to Section 6.

2. Performs the message decryption steps from [JOSE-JWE] to generate the plaintext forwarding envelope string M', using the following inputs:

   * The JWE Header H from the <header/> element's character data content.

   * The JWE Encrypted Key from the <cmk/> element's character data content.

   * The JWE Initialization Vector/Nonce from the <iv/> element's character data content.

   * The JWE Ciphertext C from the <data/> element's character data content.

   * The JWE Integrity Value I from the <mac/> element's character data content.

3. Converts the forwarding envelope UTF-8 ecoded string M' into XML element M.

4. Obtains the UTC date and time N from the <delay/> child element, and verifies it is within the accepted range, as specified in Section 7.

5. Obtains the plaintext stanza S, which is a child element node of M; the stanza MUST be fully qualified with proper namespace declarations for XMPP stanzas, to help distinguish it from other content within M.

.

5.3. Insufficient Information

At step 1, if the receiving agent is unable to obtain the CMK, or the receiving agent could not otherwise determine the additional information, it MAY return a <bad-request/> error to the sending agent (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <insufficient-information/>:

```
<message xmlns='jabber:client'
         from='juliet@capulet.lit/balcony'
         id='fJZd9WFIIwNjFctT'
         to='romeo@montegue.lit/garden'
         type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
       id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <cmk>[XML character data]</cmk>
    <iv>[XML character data]</iv>
    <data>[XML character data]</data>
    <mac>[XML character data]</mac>
  </e2e>
  <error type='modify'>
    <bad-request
        xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <insufficient-information
        xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'/>
  </error>
</message>
```

In addition to returning an error, the receiving agent 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).

5.4. Failed Decryption

At step 2, if the receiving agent is unable to successfully decrypt the stanza, the receiving agent SHOULD return a <bad-request/> error to the sending agent (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <decryption-failed/> (previously defined in [RFC3923]):

```
<message xmlns='jabber:client'
         from='juliet@capulet.lit/balcony'
         id='fJZd9WFIIwNjFctT'
         to='romeo@montegue.lit/garden'
         type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
       id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <cmk>[XML character data]</cmk>
    <iv>[XML character data]</iv>
    <data>[XML character data]</data>
    <mac>[XML character data]</mac>
  </e2e>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <decryption-failed xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'/>
  </error>
</message>
```

In addition to returning an error, the receiving agent 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).

5.5. Timestamp Not Acceptable

At step 4, if the stanza is successfully decrypted but the timestamp fails the checks outlined in Section 7, the receiving agent MAY return a <not-acceptable/> error to the sender (as described in [RFC6120]), optionally supplemented by an application-specific error condition element of <bad-timestamp/> (previously defined in [RFC3923]):

```
<message xmlns='jabber:client'
         from='juliet@capulet.lit/balcony'
         id='fJZd9WFIIwNjFctT'
         to='romeo@montegue.lit/garden'
         type='chat'>
  <e2e xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
       id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>[XML character data]</header>
    <cmk>[XML character data]</cmk>
    <iv>[XML character data]</iv>
    <data>[XML character data]</data>
    <mac>[XML character data]</mac>
  </e2e>
  <error type='modify'>
    <bad-request xmlns='urn:ietf:params:xml:ns:xmpp-stanzas'/>
    <bad-timestamp xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'/>
  </error>
</message>
```

5.6. Successful Decryption

If the receiving agent successfully decrypted the payload, it MUST NOT return a stanza error.

If the payload is an <iq/> of type "get" or "set", and the response to this <iq/> is of type "error", the receiving agent MUST send the encrypted response wrapped in an <iq/> of type "result", to prevent exposing information about the payload.

6. Requesting Session Keys

Because of the dynamic nature of XMPP stanza routing, the protocol does not exchange session keys as part of the encrypted stanza. Instead, a separate protocol is used by receiving agents to request a particular session key from the sending agent.

6.1. Request Process

Before a SMK can be requested, the receiving agent MUST have at least one public key for which it also has the private key.

To request a SMK, the receiving agent performs the following:

1. Constructs a [JOSE-JWK] JWK Set (KS), containing information about each public key the requesting agent wishes to use. Each key SHOULD include a value for the property 'kid' which uniquely

identifies it within the context of all provided keys. Each key MUST include a value for the property 'kid' if any two keys use the same algorithm.

2. Constructs a <keyreq/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace as follows:

   * The attribute 'id' set to the SMK identifier value SID.

   * The child element <pkey/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as KS, encoded base64url as pre [RFC4648].

3. Sends the <keyreq/> element as the payload of an <iq/> stanza with the attribute 'type' set to "get", the attribute 'to' set to the full JID of the original encrypted stanza's sender, and the attribute 'id' set to an opaque string value the receiving agent uses to track the <iq/> response.

6.2. Accept Process

If the sending agent approves the request, it performs the following steps:

1. Generate a JSON Web Key (JWK) representing the SMK (according to [JOSE-JPSK]):

   * The "kty" parameter MUST be "oct".

   * The "kid" parameter MUST be the SID.

   * The "k" parameter MUST be the SMK, encoded as base64url.

   * The "use" parameter, if present, MUST be set to the algorithm in use for encrypting messages from Section 4.

   * The "use" parameter, if present, MUST be set to "enc".

2. Chooses a key (PK) from the keys provided via KS, and notes its identifier value 'kid'.

3. Protects the SMK using the process outlined in [JOSE-KEYPROTECT] to generate the JWE Header (H), JWE Encrypted Key (E), JWE Initialization Vector (IV), JWE Ciphertext (C), and JWE Integrity Value (I); using the following inputs:

   * The 'alg' property is set to an algorithm appropriate for the chosen PK (e.g., "RSA-OAEP" for a "RSA" key).

   * The 'enc' property is set to the intended content encryption algorithm.

   * A randomly generated CMK. See [RFC4086] for considerations on generating random values.

   * A randomly generated initialization vector. See [RFC4086] for considerations on generating random values.

   * SMK, formatted as a JWK as above.

4. Constructs a <keyreq/> element qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace as follows:

   * The attribute 'id' set to the SMK identifier SID.

   * The child element <header/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as H, encoded base64url as per [RFC4648].

   * The child element <cmk/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as E, encoded base64url as per [RFC4648].

* The child element <iv/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as IV, encoded base64url as per [RFC4648].

* The child element <data/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as C, encoded base64url as per [RFC4648].

* The child element <mac/> qualified by the "urn:ietf:params:xml:ns:xmpp-e2e:5" namespace and with XML character data as I, encoded base64url as per [RFC4648].

5. Sends the <keyreq/> element as the payload of an <iq/> stanza with the attribute 'type' set to "result", the attribute 'to' set to the full JID from the request <iq/>'s 'from' attribute, and the attribute 'id' set to the value of the request <iq/>'s 'id' attribute.

6.3. Error Conditions

If the sending agent does not approve the request, it sends an <iq/> stanza of type "error" and containing the reason for denying the request:

o <forbidden/>: the key request is made by an entity that is not authorized to decrypt stanzas from the sending agent and/or for the indicated SID.

o <item-not-found/>: the requested SID is no longer valid.

o <not-acceptable/>: the key request did not contain any keys the sending agent understands.

6.4. Example of Successful Key Request

NOTE: unless otherwise indicated, all line breaks are included for readability.

To begin a key request, the receiving agent performs step 1 from Section 6.1 to generate the [JOSE-JWK]:

```
{
  "keys": [{
    "kty":"RSA",
    "kid":"romeo@montegue.lit/garden",
    "n":"vtqejkMF01h8oKEaHfHEYO0C2jM7eISbbSvNs0SNItYWO6GbjpJf
    N4ldXw2vpVRdysnwU3zk6o2_SD0YCH1WgeuI0QK1knMTDdNSXx52e1c4BTw
    hlA8iHuutTWmpBqesn1GNZmqB3jYsJOkVBYwCJtkB9APaBvk0itlRtizjCf
    1HHnau7nGStyshgu8-srxi_d8rC5TTLSB_zT1i6fP8fwDloemXOtC0U65by
    5P-1ZHxaf_bD8fpjps6gwSgdkZKMJAI0bOWZWuMpp2ntqa0wLB7Ndxb2Ijr
    eog_s5ssAoSiXDVdoswSbp36ZP-1lnCk2j-vZ4qbhaFg5bZtgt-gwQ",
    "e":"AQAB"
  }]
}
```

Then the receiving agent performs step 2 to generate the <keyreq/>:

```
<keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
        id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
  <pkey>
    eyJrZXlzIjpbeyJrdHkiOiJSU0EiLCJraWQiOiJyb21lb0Btb250ZWd1ZS5
    saXQvZ2FyZGVuIiwibiI6InZ0cWVqa01GMDFoOG9LRWFIZkhFWU8wQzJqTT
    dlSVNiYlN2TnMwU05JdFlXTzZHYmpwSmZONGxkWHcydnBWUmR5c253VTN6a
    zZvMl9TRDBZQ0gxV2dldUkwUUsxa25NVERkTlNYeDUyZTFjNEJUd2hsQThp
    SHV1dFRXbXBCcWVzbjFHTlptcUIzallzSk9rVkJZd0NKdGtCOUFQYUJ2azB
    pdGxSdGl6akNmMUhIbmF1N25HU3R5c2hndTgtc3J4aV9kOHJDNVRUTFNCX3
    pUMWk2ZlA4ZndEbG9lbVhPdEMwVTY1Ynk1UC0xWkh4YWZfYkQ4ZnBqcHM2Z
    3dTZ2RrWktNSkFJMGJPV1pXdU1wcDJudHFhMHdMQjdOZHhiMklqcmVvZ19z
    NXNzQW9TaVhEVmRvc3dTYnAzNlpQLTFsbkNrMmotdlo0cWJoYUZnNWJadGd
    0LWd3USIsImUiOiJBUUFCIn1dfQ
  </pkey>
</keyreq>
```

Then the receiving agent performs step 3 and sends the following:

```
<iq xmlns='jabber:client'
    from='romeo@montegue.lit/garden'
    id='xdJbWMA+'
    to='juliet@capulet.lit/balcony'
    type='get'>
  <keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
          id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <pkey>
      eyJrZXlzIjpbeyJrdHkiOiJSU0EiLCJraWQiOiJyb21lb0Btb250ZWd1Z
      S5saXQvZ2FyZGVuIiwibiI6InZ0cWVqa01GMDFoOG9LRWFIZkhFWU8wQz
      JqTTdlSVNiYlN2TnMwU05JdFlXTzZHYmpwSmZONGxkWHcydnBWUmR5c25
      3VTN6azZvMl9TRDBZQ0gxV2dldUkwUUsxa25NVERkTlNYeDUyZTFjNEJU
```

```
        d2hsQThpSHV1dFRXbXBCcWVzbjFHTlptcUIzallzSk9rVkJZd0NKdGtCO
        UFQYUJ2azBpdGxSdGl6akNmMUhIbmF1N25HU3R5c2hndTgtc3J4aV9kOH
        JDNVRUTFNCX3pUMWk2ZlA4ZndEbG9lbVhPdEMwVTY1Ynk1UC0xWkh4YWZ
        fYkQ4ZnBqcHM2Z3dTZ2RrWktNSkFJMGJPV1pXdU1wcDJudHFhMHdMQjdO
        ZHhiMklqcmVvZ19zNXNzQW9TaVhEVmRvc3dTYnAzNlpQLTFsbkNrMmotd
        lo0cWJoYUZnNWJadGd0LWd3USIsImUiOiJBUUFCIn1dfQ
      </pkey>
    </keyreq>
  </iq>
```

If the sending agent accepts this key request, it performs step 1 from Section 6.2 to generate JWK representation of the SMK:

```
{
  "kty":"oct",
  "kid":"835c92a8-94cd-4e96-b3f3-b2e75a438f92",
  "oct":"xWtdjhYsH4Va_9SfYSefsJfZu03m5RrbXo_UavxxeU8"
}
```

Then the sending agent performs steps 2 and 3 to generate the protected SMK:

JWE Header (before base64url encoding)

```
{
  "alg":"RSA-OAEP",
  "enc":"A256CBC+HS512",
  "kid":"romeo@montegue.lit/garden"
}
```

JWE Encrypted Key

```
UeoVeGcZP-VsLu1PVj3NNWkmmEF7H2Nl_mHWscOuT_vYn-4ub2NEnRy4dzyycgx
ny6jmRPpNiGJB6AfI4TYZvrjig5dubv4uG7phCvKYVI3uaUU58Fc9H_o-BTmNv2
rUT-RGt6YYLW97ZJp5ZcA2l-KxykcxaRYC4Sv_UOS3Kqo0sVx5u7tolE6SbMnUH
etg91Gc9pVVa1XX-wz4ZrcA6V8zf8pCtmc4WyDMFx8RYYXR_5Qvax-TzOJUL2eA
r3OQsf3KNh58WvvzcwAKTmR214QmZCxI_A5mIqoog0H0uV987P9yw1wFfsmg7z-
Y2Ed7Blp-zLOvXEQKU9FM-vjBnA
```

JWE Initialization Vector

```
eiXTO21DNqFnCTQkLAoAtA
```

JWE Ciphertext

```
e8sZiRvKLPOUjmFwOYhvrZMQYzW1yglg6mTnazJU_rF7mXTBIieNZCd7hDrlrdG
SxqqUgh6N1O2QBLygf2PtWDmHHjn1aLncx6qlGf0UOxCCXUBfBIhZgFH5YX1i3_
VSsNUDEoIKTGA21EnamOqa1A
```

JWE Integrity Value

```
WQzHj3j30Qo7VakMM42t-X1omQVGyebd3No9ZFGPQNUwEWONjIcZ89_wFBhZFdd
kc8i_qtXi-9XPmSVei3A_Jw
```

Then the sending agent performs step 4 to generate the <keyreq/> response:

```
<keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
        id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
  <header>
    eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZDQkMrSFM1MTIiLCJraWQ
    iOiJyb21lb0Btb250ZWd1ZS5saXQvZ2FyZGVuIn0
  </header>
  <cmk>
    UeoVeGcZP-VsLu1PVj3NNWkmmEF7H2Nl_mHWscOuT_vYn-4ub2NEnRy4dzy
    ycgxny6jmRPpNiGJB6AfI4TYZvrjig5dubv4uG7phCvKYVI3uaUU58Fc9H_
    o-BTmNv2rUT-RGt6YYLW97ZJp5ZcA2l-KxykcxaRYC4Sv_UOS3Kqo0sVx5u
    7tolE6SbMnUHetg91Gc9pVVa1XX-wz4ZrcA6V8zf8pCtmc4WyDMFx8RYYXR
    _5Qvax-TzOJUL2eAr3OQsf3KNh58WvvzcwAKTmR214QmZCxI_A5mIqoog0H
    0uV987P9yw1wFfsmg7z-Y2Ed7Blp-zLOvXEQKU9FM-vjBnA
  </cmk>
  <iv>
    eiXTO21DNqFnCTQkLAoAtA
  </iv>
  <data>
    e8sZiRvKLPOUjmFwOYhvrZMQYzW1yglg6mTnazJU_rF7mXTBIieNZCd7hDr
    lrdGSxqqUgh6N1O2QBLygf2PtWDmHHjn1aLncx6qlGf0UOxCCXUBfBIhZgF
    H5YX1i3_VSsNUDEoIKTGA21EnamOqa1A
  </data>
  <mac>
    WQzHj3j30Qo7VakMM42t-X1omQVGyebd3No9ZFGPQNUwEWONjIcZ89_wFBh
    ZFddkc8i_qtXi-9XPmSVei3A_Jw
  </mac>
</keyreq>
```

Then the sending agent performs step 5 and sends the following:

```
<iq xmlns='jabber:client'
    from='juliet@capulet.lit/balcony'
    id='xdJbWMA+'
    to='romeo@montegue.lit/garden'
    type='result'>
  <keyreq xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
          id='835c92a8-94cd-4e96-b3f3-b2e75a438f92'>
    <header>
      eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkEyNTZDQkMrSFM1MTIiLCJraWQ
      iOiJyb21lb0Btb250ZWd1ZS5saXQvZ2FyZGVuIn0
    </header>
    <cmk>
      UeoVeGcZP-VsLu1PVj3NNWkmmEF7H2Nl_mHWscOuT_vYn-4ub2NEnRy4dzy
      ycgxny6jmRPpNiGJB6AfI4TYZvrjig5dubv4uG7phCvKYVI3uaUU58Fc9H_
      o-BTmNv2rUT-RGt6YYLW97ZJp5ZcA2l-KxykcxaRYC4Sv_UOS3Kqo0sVx5u
      7tolE6SbMnUHetg91Gc9pVVa1XX-wz4ZrcA6V8zf8pCtmc4WyDMFx8RYYXR
      _5Qvax-TzOJUL2eAr3OQsf3KNh58WvvzcwAKTmR214QmZCxI_A5mIqoog0H
      0uV987P9yw1wFfsmg7z-Y2Ed7Blp-zLOvXEQKU9FM-vjBnA
    </cmk>
    <iv>
      eiXTO21DNqFnCTQkLAoAtA
    </iv>
    <data>
      e8sZiRvKLPOUjmFwOYhvrZMQYzW1yglg6mTnazJU_rF7mXTBIieNZCd7hDr
      lrdGSxqqUgh6N1O2QBLygf2PtWDmHHjn1aLncx6qlGf0UOxCCXUBfBIhZgF
      H5YX1i3_VSsNUDEoIKTGA21EnamOqa1A
    </data>
    <mac>
      WQzHj3j30Qo7VakMM42t-X1omQVGyebd3No9ZFGPQNUwEWONjIcZ89_wFBh
      ZFddkc8i_qtXi-9XPmSVei3A_Jw
    </mac>
  </keyreq>
</iq>
```

7. Inclusion and Checking of Timestamps

Timestamps are included to help prevent replay attacks. All timestamps MUST conform to [XEP-0082] 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 agent:

- 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 might 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 [XEP-0160], when sending an offline message to the recipient, the server SHOULD include delayed delivery data as specified in [XEP-0203] 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.

8. Interaction with Stanza Semantics

The following limitations and caveats apply:

- Undirected <presence/> stanzas SHOULD NOT be encrypted. Such stanzas are delivered to anyone the sender has authorized, and can generate a large volume of key requests.

- 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.

9. Mandatory-to-Implement Cryptographic Algorithms

All algorithms that MUST be implemented for [JOSE-JWE] also MUST be implemented for this specification.

10. Security Considerations

10.1. Storage of Encrypted Stanzas

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.

10.2. Re-use of Session Master Keys

A sender SHOULD NOT use the same SMK for stanzas intended for different recipients, as determined by the localpart and domainpart of the recipient's JID.

A sender MAY re-use a SMK for several stanzas to the same recipient. In this case, the SID remains the same, but the sending agent MUST generate a new CMK and IV for each encrypted stanza. The sender SHOULD periodically generate a new SMK; however, this specification does not mandate any specific algorithms or processes.

In the case of <message/> stanzas, a sending agent might generate a new SMK each time it generates a new ThreadID, as outlined in [XEP-0201].

11. IANA Considerations

11.1. 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-e2e:5

Specification: RFC XXXX

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

Registrant Contact: IESG, <iesg@ietf.org>

12. References

12.1. Normative References

[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.

[JOSE-JWA] Jones, M., "JSON Web Algorithms (JWA)", draft-ietf-jose-json-web-algorithms-08 (work in progress), December 2012.

[JOSE-JWE] Jones, M., Rescola, E., and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-08 (work in progress), December 2012.

[JOSE-JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-08 (work in progress), December 2012.

[JOSE-JPSK] Jones, M., "JSON Private and Symmetric Key", draft-jones-jose-json-private-and-symmetric-key-00 (work in progress), December 2012.

[JOSE-KEYPROTECT] Miller, M., "Using JSON Web Encryption (JWE) for Protecting JSON Web Key (JWK) Objects", draft-miller-jose-jwe-protected-jwk-00 (work in progress), February 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", RFC 4949, August 2007.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.

[XEP-0030]

Eatmon, R., Hildebrand, J., Millard, P., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2006.

[XEP-0082]
Saint-Andre, P., "XMPP Date and Time Profiles", XSF XEP 0082, May 2003.

[XEP-0115]
Hildebrand, J., Troncon, R., and P. Saint-Andre, "Entity Capabilities", XSF XEP 0115, February 2008.

[XEP-0203]
Saint-Andre, P., "Delayed Delivery", XSF XEP 0203, September 2009.

[XEP-0297]
Wild, M. and K. Smith, "Stanza Forwarding", XSF XEP 0297, July 2012.

12.2. Informative References

[RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3923, September 2003.

[RFC3923] Saint-Andre, P., "End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP)", RFC 3923, October 2004.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", RFC 4086, June 2005.

[XEP-0160]
Saint-Andre, P., "Best Practices for Handling Offline Messages", XSF XEP 0160, January 2006.

[XEP-0201]
Saint-Andre, P., Paterson, I., and K. Smith, "Best Practices for Message Threads", XSF XEP 0203, November 2010.

Appendix A. Schema for urn:ietf:params:xml:ns:xmpp-e2e:5

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:5'
    xmlns='urn:ietf:params:xml:ns:xmpp-e2e:5'
    elementFormDefault='qualified'>

  <xs:element name='e2e'>
    <xs:complexType>
      <xs:attribute name='id' type='xs:string' use='required'/>
      <xs:sequence>
        <xs:element ref='header' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='cmk' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='iv' minOccurs=1' maxOccurs='1'/>
        <xs:element ref='data' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='mac' minOccurs='1' maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='keyreq'>
    <xs:complexType>
      <xs:attribute name='id' type='xs:string' use='required'/>
      <xs:sequence>
        <xs:element ref='pkey' minOccurs='0' maxOccurs='1'/>
        <xs:element ref='header' minOccurs='0' maxOccurs='1'/>
        <xs:element ref='cmk' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='iv' minOccurs=1' maxOccurs='1'/>
        <xs:element ref='data' minOccurs='1' maxOccurs='1'/>
        <xs:element ref='mac' minOccurs='1' maxOccurs='1'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='cmk'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
        </xs:extension>
      </xs:simpleType>
    </xs:complexType>
  </xs:element>

  <xs:element name='iv'>
    <xs:complexType>
      <xs:simpleType>
        <xs:extension base='xs:string'>
        </xs:extension>
      </xs:simpleType>
    </xs:complexType>
  </xs:element>
```

```
    <xs:element name='data'>
      <xs:complexType>
        <xs:simpleType>
          <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

    <xs:element name='header'>
      <xs:complexType>
        <xs:simpleType>
          <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

    <xs:element name='mac'>
      <xs:complexType>
        <xs:simpleType>
          <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

    <xs:element name='pkey'>
      <xs:complexType>
        <xs:simpleType>
          <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

    <xs:element name='smk'>
      <xs:complexType>
        <xs:simpleType>
          <xs:extension base='xs:string'>
          </xs:extension>
        </xs:simpleType>
      </xs:complexType>
    </xs:element>

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

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

  </xs:schema>
```

Author's Address

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
USA

Phone: +1-303-308-3204
Email: mamille2@cisco.com