XEP-0420: Stanza Content Encryption

Paul Schaub
mailto:vanitasvitae@riseup.net
xmpp:vanitasvitae@jabberhead.tk

2020-07-03
Version 0.3.0

Status	Type	Short Name
Experimental	Standards Track	SCE

The Stanza Content Encryption (SCE) protocol is intended as a way to allow clients to securely exchange arbitrary extension elements using different end-to-end encryption schemes.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

There is a number of different end-to-end encryption mechanisms that can be used to secure user communication against unauthorized access from malicious third parties. Popular examples for this are OMEMO Encryption (XEP-0384) [1] and OpenPGP for XMPP (XEP-0373) [2]. While the latter allows for encryption of arbitrary extension elements, protocols such as OMEMO Encryption (XEP-0384) [3] are limited to only encrypt the body of a message. This approach is not very flexible and prevents the combined usage with XMPP extension protocols such as Stateless Inline Media Sharing (XEP-0385) [4] or Last Message Correction (XEP-0308) [5] as their extension elements cannot be included in the encrypted part of the message, therefore leaking information about the message content.
This extension protocol proposes a solution to aforementioned issues by generalizing the OpenPGP Content Elements (eg. <signcrypt>) introduced by OpenPGP for XMPP (XEP-0373) [6] for the use with other encryption protocols.

2 Requirements

This proposal widens the scope of the security guarantees given by the used encryption mechanism from just the body of the message to all contents of the <content/> element. It is intended to serve as a ”one size fits all” solution for extension element encryption in XMPP.
In order to achieve its goal, Stanza Content Encryption does the following:

- Define elements that hold sensitive information
- Speficy rules about how extension elements are encrypted and embedded in the message
- Specify rules about which elements are allowed inside and outside the protected domain

3 Glossary

Envelope Element <envelope/> An XMPP extension element which is used to hold the encrypted <content/> element.

Content Element <content/> An element which is used to contain all of those extension elements that need to be encrypted. The XML representation of this element is encrypted and then embedded into the <envelope/> element.

[1] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[2] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.
[3] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[4] XEP-0385: Stateless Inline Media Sharing (SIMS) <https://xmpp.org/extensions/xep-0385.html>.
[5] XEP-0308: Last Message Correction <https://xmpp.org/extensions/xep-0308.html>.
[6] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.

4 Affix Elements

In order to prevent certain attacks, different affix elements MAY be added into the <content/> element.

Element	Description	Usage	Verification
<rpad/>	Random-length random-content padding	Prevent known ciphertext and message length correlation attacks. The content of this element is a randomly generated sequence of random length between 0 and 200 characters. TODO: sane boundaries?	None. This element is only used to change the length of the ciphertext and doesn't need to be verified
<time/>	Timestamp	Prevent replay attacks using old messages. This element MUST have one attribute 'stamp', whos value is a timestamp following the format described in XMPP Date and Time Profiles (XEP-0082) XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.. The timestamp represents the time at which the message was encrypted by the sender.	Receiving clients MUST check whether the difference between the timestamp and the sending time derived from the stanza itself lays within a reasonable margin. The client SHOULD use the content of the timestamp element when displaying the send date of the message
<to/>	Recipient of the message	Prevent spoofing of the recipient. This element MUST have one attribute 'jid', whos value is the JID of the intended recipient.	Receiving clients MUST check, if the JID matches the to attribute of the enclosing stanza and otherwise alert the user/reject the message
<from/>	Sender of the message	Prevent spoofing of the sender. This element MUST have one attribute 'jid', whos value is the JID of the sender of the message.	Receiving clients MUST check, if the value matches the from attribute of the enclosing stanza and otherwise alert the user/reject the message

Listing 1: Examples of Affix Elements

```
<time stamp='2004-01-25T06:05:00+01:00'/>
<to jid='missioncontrol@houston.nasa.gov'/>
<from jid='opportunity@mars.planet'/>
<rpad>C1DHN9HK-9A25tSmwK4hU!Jji9%GKYK^syIlHJT9TnI4</rpad>
```

Encryption protocols that make use of Stanza Content Encryption MUST define their own profiles that describe mandatory behaviour of which of these elements are used. They MAY also define and add their own specific affix elements.

5 Motivation

Some end-to-end encryption protocols like OMEMO Encryption (XEP-0384) [7] are historically limited to encryption of the message body only. This approach excludes other extension elements from the protected domain of the payload element, exposing them to potential attackers.

Listing 2: An imperfectly encrypted message which leaks dangerous information about the conversation through the plaintext OOB extension element

```
<message from='narrator@jabber.org'
         to='viewer@jabber.org'>
  <encrypted xmlns='eu.siacs.conversations.axolotl'>
    <header sid='27183'>
      ...
    </header>
    <payload>
      SSBnb3QgaW4gZXZlcnlvbmUncyBob3N0aWxlIGxpdHRsZSBmYWNlLiBZZXMsIHRoZXNlIGFyZSBi

      cnVpc2VzIGZyb20gZmlnaHRpbmcuIFllcywgSSdtIGNvbWZvcnRhYmxlIHdpdGggdGhhdC4gSSBh

      bSBlbmxpZ2h0ZW5lZC4=
    </payload>
  </encrypted>
  <x xmlns='jabber:x:oob'>
    <url>https://en.wikipedia.org/wiki/Fight_Club#Plot</url>
  </x>
</message>
```

The example above obviously leaks information about the communication through the unencrypted OOB extension element.

[7] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.

Most end-to-end encryption mechanisms are also focussed solely on message content encryption and do not tackle <iq/> requests/replies at all. Stanza Content Encryption can be applied to those as well.

Listing 3: Unencrypted IQ request

```
<iq from='doctor@shakespeare.lit/pda'
    id='get-data-1'
    to='ladymacbeth@shakespeare.lit/castle'
    type='get'>
  <data xmlns='urn:xmpp:bob'
        cid='sha1+8f35fef110ffc5df08d579a50083ff9308fb6242@bob.xmpp.
            org'/>
</iq>
```

Listing 4: Likewise unencrypted reply

```
<iq from='ladymacbeth@shakespeare.lit/castle'
    id='get-data-1'
    to='doctor@shakespeare.lit/pda'
    type='result'>
  <data xmlns='urn:xmpp:bob'
        cid='sha1+8f35fef110ffc5df08d579a50083ff9308fb6242@bob.xmpp.
            org'
        max-age='86400'
        type='image/png'>
    iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKCAYAAACNMs+9AAAABGdBTUEAALGP
    C/xhBQAAAAlwSFlzAAALEwAACxMBAJqcGAAAAAd0SU1FB9YGARc5KB0XV+IA
    AAAddEVYdENvbW1lbnQAQ3JlYXRlZCB3aXRoIFRoZSBHSU1Q72QlbgAAAF1J
    REFUGNO9zL0NglAAxPEfdLTs4BZM4DIO4C7OwQg2JoQ9LE1exdlYvBBeZ7jq
    ch9//q1uH4TLzw4d6+ErXMMcXuHWxId3KOETnnXXV6MJpcq2MLaI97CER3N0
    vr4MkhoXe0rZigAAAABJRU5ErkJggg==
  </data>
</iq>
```

6 Use Cases

6.1 Use in <message/> stanzas

The main use case of Stanza Content Encryption is the use of end-to-end encryption protocols in combination with extension protocols that store sensitive information in other places than the message body.
This applies to many extension elements that add additional information to <message/> stanzas, such as those of Out-of-Band Data (XEP-0066) [8].

[8] XEP-0066: Out of Band Data <https://xmpp.org/extensions/xep-0066.html>.

Listing 5: Content element containing the messages body and the OBB element.

```
<content xmlns='urn:xmpp:sce:0'>
  <payload>
    <body xmlns='jabber:client'>[...]</body>
    <x xmlns='jabber:x:oob'>
      <url>https://en.wikipedia.org/wiki/Fight_Club#Plot</url>
    </x>
  </payload>
</content>
```

Listing 6: Finished message stanza containing the <content/> element from the previous example encrypted using a hypothetical encryption protocol and SCE.

```
<message from='narrator@jabber.org'
         to='viewer@jabber.org'>
  <encrypted xmlns='urn:xmpp:encryption:stub:sce:0'>
    <payload>
      PGNvbnRlbnQgeG1sbnM9J3Vybjp4bXBwOnNjZTowJz48cGF5bG9hZD48Ym9keSB4bWxucz0namFi

      YmVyOmNsaWVudCc+
          SSBnb3QgaW4gZXZlcnlvbmUncyBob3N0aWxlIGxpdHRsZSBmYWNlLiBZZXMs
      IHRoZXNlIGFyZSBicnVpc2VzIGZyb20gZmlnaHRpbmcuIFllcywgSSdtIGNvbWZvcnRhYmxlIHdp

      dGggdGhhdC4gSSBhbSBlbmxpZ2h0ZW5lZC48L2JvZHk+
          PHggeG1sbnM9J2phYmJlcjp4Om9vYic+
      PHVybD5odHRwczovL2VuLndpa2lwZWRpYS5vcmcvd2lraS9GaWdodF9DbHViI1Bsb3Q8L3VybD48

      L3g+PC9wYXlsb2FkPjwvY29udGVudD4=
    </payload>
  </encrypted>
</message>
```

6.2 Use in <iq/> stanzas

Stanza Content Encryption thrives not only to allow for rich content encryption in <message/> stanzas, but is also applicable to <iq/> queries. A resource might want to query sensitive information from another resource capable of Stanza Content Encryption.

Listing 7: Sender prepares a <content/> element containing the query subject.

```
<content xmlns='urn:xmpp:sce:0'>
  <payload>
    <data xmlns='urn:xmpp:bob'
        cid='sha1+8f35fef110ffc5df08d579a50083ff9308fb6242@bob.xmpp.
           org'/>
  </payload>
  <from jid='doctor@shakespeare.lit/pda'/>
  <to jid='ladymacbeth@shakespear.lit/castle'/>
```

```
</content>
```

Listing 8: The sender then encrypts the <content/> element for the recipient and sends the <iq/> containing the result of the encryption.

```
<iq from='doctor@shakespeare.lit/pda'
    id='get-data-1'
    to='ladymacbeth@shakespeare.lit/castle'
    type='get'>
  <encrypted xmlns='urn:xmpp:encryption:stub:sce:0'>
    <payload>
      V2FpdCwgd2hhdD8gQXJlIHlvdSBzZXJpb3VzPyBEaWQgeW91IHJlYWxseSBqdXN0IGdyYWIgeW91

      ciBmYXZvdXJpdGUgYmFzZTY0IGRlY29kZXIganVzdCB0byBjaGVjayB0aGlzIGRvY3VtZW50IGZv

      ciBoaWRkZW4gbWVzc2FnZXM/
         IFdoYXQgYXJlIHlvdSBzb21lIGtpbmQgb2YgbmVyZD8gU29tZSBn
      ZWVrIHdpdGggYSBiaW5hcnkgd3Jpc3Qgd2F0Y2g/
    </payload>
  </encrypted>
</iq>
```

Listing 9: The recipient prepares the reply to the request by assembling the <content/> element.

```
<content xmlns='urn:xmpp:sce:0'>
  <payload>
    <data xmlns='urn:xmpp:bob'
        cid='sha1+8f35fef110ffc5df08d579a50083ff9308fb6242@bob.xmpp.
           org'
        max-age='86400'
        type='image/png'>
    iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKCAMAAAC67D+
       PAAAAclBMVEUAAADYZArfaA9GIAoBAAGN
    QA3MXgniaAiEOgZMIATDXRXZZhHUZBHIXhDrbQ6sUQ7OYA2TRAubRwqMQQq7VQlKHgMAAAK5WRfJ

    YBOORBFoMBCwUQ/ycA6FPgvbZQpeKglNJQmrTQeOPgQyFwR6MwACAABRPE/
       oAAAAW0lEQVQI1xXI
    Rw6EMBTAUP8kJKENnaF37n9FQPLCekAgzklhgCwfrlNHEXhrvCsxaU/
       SwLGAFuIWZFpBERtKm9Xf
    JqH+vVWh4POqgHrsAtht095b+geYRSl57QHSPgP3+CwvAAAAAABJRU5ErkJggg==
    </data>
  </payload>
  <from jid='ladymacbeth@shakespear.lit/castle'/>
  <to jid='doctor@shakespeare.lit/pda'/>
</content>
```

Listing 10: The <content/> element is then encrypted and sent as a reply to the initiator of the request.

```
<iq from='ladymacbeth@shakespeare.lit/castle'
    id='get-data-1'
    to='doctor@shakespeare.lit/pda'
    type='result'>
  <encrypted xmlns='urn:xmpp:encryption:stub:sce:0'>
    <payload>
      PGNvbnRlbnQgeG1sbnM9J3Vybjp4bXBwOnNjZTowJz4KICA8cGF5bG9hZD4KICAgIDxkYXRhIHht

      bG5zPSd1cm46eG1wcDpib2InCiAgICAgICAgY2lkPSdzaGExKzhmMzVmZWYxMTBmZmM1ZGYwOGQ1

      NzlhNTAwODNmZjkzMDhmYjYyNDJAYm9iLnhtcHAub3JnJwogICAgICAgIG1heC1hZ2U9Jzg2NDAw

      JwogICAgICAgIHR5cGU9J2ltYWdlL3BuZyc+
          CiAgICBpVkJPUncwS0dnb0FBQUFOU1VoRVVnQUFB
      QW9BQUFBS0NBTUFBQUM2N0QrUEFBQUFjbEJNVkVVQUFBRFlaQXJmYUE5R0lBb0JBQUdOCiAgICBR

      QTNNWGduaWFBaUVPZ1pNSUFURFhSWFpaaEhVWkJISVhoRHJiUTZzVVE3T1lBMlRSQXViUndxTVFR

      cTdWUWxLSGdNQUFBSzVXUmZKCiAgICBZQk9PUkJGb01CQ3dVUS95Y0E2RlBndmJaUXBlS2dsTkpR

      bXJUUWVPUGdReUZ3UjZNd0FDQUFCUlBFL29BQUFBVzBsRVFWUUkxeFhJCiAgICBSdzZFTUJUQVVQ

      OGtKS0VObmFGMzduOUZRUExDZWtBZ3prbGhnQ3dmcmxOSEVYaHJ2Q3N4YVUvU3dMR0FGdUlXWkZw

      QkVSdEttOVhmCiAgICBKcUgrdlZXaDRQT3FnSHJzQXRodDA5NWIrZ2VZUlNsNTdRSFNQZ1AzK0N3

      dkFBQUFBQUJKUlU1RXJrSmdnZz09CiAgICA8L2RhdGE+
          CiAgPC9wYXlsb2FkPgogICAgIDxmcm9tIGpp
      ZD0nbGFkeW1hY2JldGhAc2hha2VzcGVhci5saXQvY2FzdGxlJy8+
          CiAgPHRvIGppZD0nZG9jdG9y
      QHNoYWtlc3BlYXJlLmxpdC9wZGEnLz4KPC9jb250ZW50Pgo=
    </payload>
  </encrypted>
</iq>
```

7 Sending an encrypted stanza

In order to send an encrypted message without leaking extension elements the sender prepares the message by placing the sensitive extension elements inside a <payload/> element inside a <content/> element.
Depending on the encryption-specific SCE-profile, some affix elements are added as child elements of the <content/> element.
The <content/> element is then serialized into XML and encrypted using the SCE-specific profile of the encryption mechanism in place. The result is appended to the message.
Since the outer message element does not contain a <body/> element the sender appends an

unencrypted <store/> hint as specified in Message Processing Hints (XEP-0334) [9].
The message can then be sent to the recipient.

8 Receiving an encrypted stanza

The recipient of the message decrypts the content of the <envelope/> element to retrieve the <content/> element. Depending on the affix profiles specified by the used encryption protocol, the affix elements are verified to prevent certain attacks from taking place.
Next the extension elements of the <content/> elements <payload/> element are checked against the whitelist/blacklist and any disallowed elements are discarded.
As a last step, the original unencrypted stanza is recreated by replacing the <envelope/> element of the stanza with the contents of the <payload/> element.

9 Server-processed Elements

There are certain extension elements which are required to be available to the server in order to do message routing and processing. Additionally there are some elements that MUST be filtered by the server. Allowing for those elements to be included in, and parsed from the encrypted payload would allow a malicious client to perform a number of attacks.
Contrary to this, other elements are considered sensitive and MUST NOT be available in plaintext outside the <content/> element.
It is hard to come up with a complete list of exceptional elements at this point, as there is no practical implementation experience.
Below is a non-exhaustive list of elements that are definitely blacklisted inside the <content/> element and whitelisted as direct child elements of the message.

[9] XEP-0334: Message Processing Hints <https://xmpp.org/extensions/xep-0334.html>.

Element
Elements of Message Processing Hints (XEP-0334) XEP-0334: Message Processing Hints <https://xmpp.org/exter

Element
Stanza-ID elements of Unique and Stable Stanza IDs (XEP-0359) XEP-0359: Unique and Stable Stanza IDs <https:/

Element
Elements of Extended Stanza Addressing (XEP-0033) XEP-0033: Extended Stanza Addressing <https://xmpp.org/

TODO: Other elements?

10 Business Rules

Unencrypted <content/> elements are NOT ALLOWED as child elements of the stanza and MUST be dropped.

Elements in the <content/> elements <payload/> element MUST be identified using an element name and namespace. Notably the <body/> element MUST contain a valid namespace (i.e. "jabber:client").
The recipient must verify that the decrypted <content/> element contains valid XML before processing it any further. Invalid XML must be rejected.
After verifying the integrity of the <content/> element, the recipient needs to make sure that no server-processed elements are found within the payload. Any forbidden elements MUST be dropped before the message is processed any further.
Furthermore the receiving client MUST ignore any extension elements considered as sensitive which are found outside of the <content/> element, especially as direct unencrypted child elements of the enclosing stanza.
Since a chat message encrypted with SCE MUST NOT contain a <body/> element, it is not eligible for MAM message storage (Message Archive Management (XEP-0313) [10]). Therefore sending entities MUST append an unencrypted Message Processing Hints (XEP-0334) [11] <store/> hint as a direct child element to the message.

11 Implementation Notes

As a first, naïve approach a recipient of a message containing an <envelope/> element could simply reinject the reassambled unencrypted stanza into the XML stream. This might introduce some security issues. Most notably, depending on the clients implementation it may become ambiguous which elements were received end-to-end encrypted and which were received unencrypted.
Implementations should rather handle encrypted elements explicitly.

12 Security Considerations

For the sake of simplicity, the examples in this document are not encrypted. A real-world implementation MUST make use of real cryptographic protocols.

12.1 Encryption Profiles

This specification presents a set of affix elements which can be used to counter certain attacks. However it does not dictate any behaviour regarding what elements MUST be used/verified or when.
Different cryptographic protocols come with different possible attack scenarios which must be taken into consideration, so it is left up to those cryptographic protocols to define profiles

[10] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[11] XEP-0334: Message Processing Hints <https://xmpp.org/extensions/xep-0334.html>.

that describe the use of affix elements.

13 XMPP Registrar Considerations

TODO: Maybe the Registrar should handle a blacklist of elements that are allowed as child elements of the <content/> element?

14 XML Schema

TODO.

15 Acknowledgements

Big thanks to the authors of OpenPGP for XMPP (XEP-0373) [12] (Florian Schmaus, Dominik Schürmann and Vincent Breitmoser) which heavily inspired the idea of this protocol.
Also thanks to Marvin Wißfeld, Tim Henkes, Daniel Gultsch, Melvin Keskin and Andreas Straub for their feedback.

[12] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.