XEP-0420: Stanza Content Encryption

Paul Schaub
mailto:vanitasvitae@fsfe.org
xmpp:vanitasvitae@jabberhead.tk

2019-07-30
Version 0.1.0

<table>
<thead>
<tr>
<th>Status</th>
<th>Type</th>
<th>Short Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental</td>
<td>Standards Track</td>
<td>SCE</td>
</tr>
</tbody>
</table>

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

This XMPP Extension Protocol is copyright © 1999 – 2018 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

## NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. ##

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

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).
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)** and **OpenPGP for XMPP (XEP-0373)**.

While the latter allows for encryption of arbitrary extension elements, protocols such as **OMEMO Encryption (XEP-0384)** 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)** or **Last Message Correction (XEP-0308)** 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)** 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 and their encrypted form
- Specify 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.

---

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Usage</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;rpad/&gt;</code></td>
<td>Random-length random-content padding</td>
<td>Prevent known ciphertext and message length correlation attacks</td>
<td>None. This element is only used to change the length of the ciphertext and doesn't need to be verified.</td>
</tr>
<tr>
<td><code>&lt;time/&gt;</code></td>
<td>Timestamp</td>
<td>Prevent replay attacks using old messages</td>
<td>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.</td>
</tr>
<tr>
<td><code>&lt;to/&gt;</code></td>
<td>Recipient of the message</td>
<td>Prevent spoofing of the recipient</td>
<td>Receiving clients MUST check, if they are recipient of the message and otherwise alert the user/reject the message.</td>
</tr>
<tr>
<td><code>&lt;from/&gt;</code></td>
<td>Sender of the message</td>
<td>Prevent spoofing of the sender</td>
<td>Receiving clients MUST check, if the content of the from element matches the from attribute of the enclosing stanza and otherwise alert the user/reject the message.</td>
</tr>
</tbody>
</table>

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 Use Cases

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. Some end-to-end encryption protocols like OMEMO Encryption (XEP-0384) 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 1: The Narrator sends an imperfectly encrypted message, leaking dangerous information about the conversation through the OOB extension element.

```xml
<message from='narrator@jabber.org' to='viewer@jabber.org'>
  <encrypted xmlns='eu.siacs.conversations.axolotl'>
    <header sid='27183'>
      ...
    </header>
    <payload>
      SSBnb3QgaW4gZXZlcnlvbmUncyBob3N0aWxlIGxpHRsZSBmYWNIhIHZRZXN1IGFyZSBicnVpc2VzIGZyb20gZmlnaHRpbmcucHJlc2NhbmdpbmciLCJuZXRlcyIgZm9yIHNvbW1lbnN0IGZlcnNpb24gaW4gYm9keiBzdHlsZSBzaG9wZS5i
      bSBibmxpZ2h0ZW5lZC4=
    </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. Another example of a possible use case would be encrypted <iq/> queries. A resource might want to query sensitive information from another resource capable of Stanza Content Encryption. This problem can also be solved using SCE.

6 Sending an encrypted message

To fix the issue of leaking extension elements using Stanza Content Encryption, the sender prepares the message by placing the content elements inside the <content/> element.

Listing 2: Content element containing the messages body and the OBB element.

```
```
The `<content/>` element is then serialized into XML and encrypted using the encryption mechanism in place. The result is embedded in the `<envelope/>` element and appended to the message.

Listing 3: Same message as in the previous example but encrypted using Stanza Content Encryption to also protect the OBB extension element.

```
<message from='narrator@jabber.org' to='viewer@jabber.org'>
  <encrypted xmlns='eu.siacs.conversations.axolotl'>
    <header sid='27183'>
      ...
    </header>
    <envelope xmlns='urn:xmpp:sce:0'>
      PGNvbnRIbnQgeGlsbnM9J3Vyb4pXWw0nNjZTowJz48cGF5b9hZD48Ym9keSB46Wxucz0namFi
      YmVyOmNsaWVudC+SSBnb3QgaW4gZXZlcn1vbUcCybo3N0aWx1IGxpdrZbYWNlLiBZZXMs
      IHRoZXN1IGFyZSBicnVpc2VzIGZyB20gZm1naHRpbmcuIFlcywSSdtIGNvbWZvcmRhYmxlIHp
      dGggdGhhdC4gSSBhbSB1bmxpZ2h0ZWR5L3g8L3g+PC9wYXlsb2FkPjwvY29udGVudD4=
    </envelope>
  </encrypted>
</message>
```

The message can then be sent to the recipient.

### 7 Receiving an encrypted message

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.

8 Blacklist

The receiving client MUST ignore certain elements that may allow for attacks to take place. Since it is hard to come up with a complete list of blacklisted elements, a general rule of thumb would be the following:

Blacklisted are all elements that need to be read by the server at some point.

Below is an additional list of elements that are definitely forbidden inside the `<content/>` element and MUST instead be placed in the message unencrypted.

<table>
<thead>
<tr>
<th>Element</th>
<th>Reason</th>
</tr>
</thead>
</table>
9 Business Rules

Unencrypted <content/> elements are NOT ALLOWED as child elements of the stanza and MUST be dropped. Elements in the <content/> element MUST be identified using an element name and namespace. Notably the <body/> element MUST contain a valid namespace (i.e. "jabber:client"). 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 blacklisted elements are found within the payload. Any forbidden elements MUST be dropped before the message is processed any further.

Duplicate elements within the <content/> element MUST be dropped.

Elements in the <content/> element override elements in the enclosing stanza. TODO: Maybe we want to remove this rule by disallowing duplicate elements all together?

TODO: Other elements?
10 Implementation Notes

As a first, naïve approach a recipient of a message containing an `<envelope/>` element could simply reinject the reassembled unencrypted stanza into the XML stream. This might introduce some security issues. Most notably, there is no way to distinguish end-to-end encrypted elements from unencrypted elements. Implementations should rather handle encrypted elements explicitly.

11 Security Considerations

11.1 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 that describe the use of affix elements.

12 XMPP Registrar Considerations

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

13 XML Schema

TODO.