XEP-0473: OpenPGP for XMPP Pubsub

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr

2022-12-13
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	oxps

Specifies an OpenPGP for XMPP (XEP-0373) profile for the pubsub use case.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Publish-Subscribe (XEP-0060) [1] and Personal Eventing Protocol (XEP-0163) [2] are widely used XMPP features. However, items are published in plain text, making them readable by server administrators or anybody having administrator level access on the pubsub/PEP services.
Although this is not a problem for data intended to be published publicly, there are many cases where it may be desirable to make the data inaccessible to anybody but the selected recipients, or in other words to end-to-end encrypt the data.
This specification aims to provide an easy way to e2e encrypt items of a pubsub node, and to share access with other entities. It is compatible with all existing pubsub/PEP features, which makes it possible to use securily features such a geolocation, private blogs, private calendar events, data forms, etc.

2 Requirements

The design goals of this XEP are:

- work with basic standard pubsub/PEP service, no extra feature needed;
- be generic and work with any protocol using pubsub: once decrypted, the items should be parsable as if they were plain text;
- be able to share with other entities the means to decipher the elements;
- be able to easily share the means of decrypting items with new entities, even after the items have been published;
- be able to modify the means of decryption to make the new elements inaccessible to certain entities even if they have had access to the previous elements;

3 Glossary

- **e2ee**: end-to-end encryption
- **Shared Secret**: a cryptographically strong random string used to symmetrically encrypt items
- **Encrypted Node**: pubsub node whose items are end-2-end encrypted using this protocol.

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

4 Overview

To encrypt a pubsub node, each item is symmetrically encrypted with a shared secret using OpenPGP symmetric encryption. The secret is shared with other entities using e2e encrypted messages with OpenPGP for XMPP (XEP-0373) [3]. If an entity's access is revoked or a shared secret may have been compromised, a new shared secret is generated and new items are encrypted with it.

5 Use Cases

5.1 Encrypting Pubsub Item

Juliet wants to create a private blog (using Microblogging Over XMPP (XEP-0277) [4]) that she only share with her confidante and her lover. To make sure that her family, who manages her XMPP server, can't read the content, she want to use end-to-end encryption.
To do so, her client creates a "shared secret" by generating a cryptographically strong key that will be used to symmetrically encrypt new items. This key MUST be associated with an ID, which MUST be an unique sequence of characters usable in an XML attribute. She will later share this key with entities allowed to access or publish on the blog as explained below.
From now on, all items that Juliet publishes on the node SHOULD be symmetrically encrypted with the shared secret by using OpenPGP symmetric encryption.
To publish an encrypted item, an <encrypted/> element qualified by the 'urn:xmpp:openpgp:pubsub:0' namespace MUST be used as payload. This element MUST contain a 'secret' attribute whose value is the ID of the shared secret used. The content of the element is a base64 encoded Symmetric-Key Encrypted Session Key Packet as specified at RFC 4880 [5] § 5.3 (in a similar way as secret key backup is encoded in XEP-0373). The encrypted content is the item payload that would normally be used.
The encrypted node SHOULD have a "whitelist" access model as specified in Publish-Subscribe (XEP-0060) [6]. Juliet's client ensure that either by creating a new node with suitable access model, or by changing the access model of existing node.

Listing 1: Juliet publish an encrypted item

```
<iq
  type='set'
  from='juliet@capulet.lit/chamber'
  to='pubsub.capulet.lit'
  id='encrypted1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='123abc'>
```

[3] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.
[4] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[5] RFC 4880: OpenPGP Message Format <http://tools.ietf.org/html/rfc4880>.
[6] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

```
      <item id="encrypt_123">
        <encrypted xmlns='urn:xmpp:openpgp:pubsub:0' key='1234-abcd
            -5678-efgh'>
          <!-{}- BASE64_OPENPGP_SYMMETRICALLY_ENCRYPTED_ITEM -{}->
        </encrypted>
      </item>
    </publish>
  </pubsub>
</iq>
```

5.2 Transmit Shared Secret

Now that Juliet has started to publish encrypted items on a blog, she wants to share the secret key with her confidante and her lover. To do so she uses a e2ee <message/> stanza by using an <openpgp/> element wrapping a <signcrypt/> as specified in XEP-0373 "Exchanging OpenPGP Encrypted and Signed Data". The encrypted payload of the <signcrypt/> MUST contain a <shared-secret/> element qualified by the 'urn:xmpp:openpgp:pubsub:0' namespace. This element MUST have the following attributes:

- a 'timestamp' attribute whose value is the datetime of the shared key generation. Datetime format is specified in XMPP Date and Time Profiles (XEP-0082) [7].
- a 'jid' attribute with bare JID of the pubsub/PEP service;
- a 'node' attribute with the name of the encrypted node;
- an 'id' attribute with the ID of the shared secret;
- if the key is revoked (due to shared secret rotation), it must have a 'revoked' attribute with the value "true".

The <shared-secret/> SHOULD have a 'type' attribute whose value is the pubsub#type that the node would have if it were plain text (i.e. the semantic type information of decrypted data in the node, usually the namespace of the decrypted payload).
The content of the <shared-secret/> element MUST be the shared secret (i.e. the passphrase used to symmetrically encode items).
She send messages with the encrypted shared secret to her nurse, Romeo and herself (so her other devices get the shared secret too).
Entities MUST always use the most recently generated shared secret to encrypt new items (the 'timestamp' attribute is used to check generation date and time).
Following example only show the <shared-secret/> element, as it is normally put as an encrypted payload of OpenPGP for XMPP (XEP-0373) [8] <signcrypt/> payload.

[7] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.
[8] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.

Listing 2: shared-secret element (normally encrypted as part of XEP-0373's signcrypt payload)

```
<shared-secret
    xmlns='urn:xmpp:openpgp:pubsub:0'
    jid='pubsub.capulet.lit'
    node='123abc'
    id='1234-abcd-5678-efgh'
    timestamp='2022-10-10T13:24:31Z'
    type='http://www.w3.org/2005/Atom'
    >
    ZSRD5lK9mz-5VHNyu2N1XLiJZ8I87jkv85ceZkVrOGA
  </shared-secret>
```

note: we use <message/> instead of PEP or Pubsub to transmit the shared secret for several reasons:

- Shared secret would have to be encrypted for all entities allowed to decrypt the items, and re-encrypted each time there is a new entity. While acceptable for a small number of entities, this doesn't scale as well as separated <message/>;
- The item containing the shared secret would need to keep all previous shared secrets in case of secret rotation (see below). At some point, there would be a risk to reach stanza maximum size limit of one server;
- Using a well-known pubsub node or pubsub item id would mean than the location of the encrypted shared secret would be known. While not enough to get the secret, that's better to hide it in encrypted messages flow;

5.3 Revocation and Shared Secret Rotation

If there is any doubt about the compromise of a shared secret or if access to the encrypted node is revoked for an entity, the shared secret SHOULD BE rotated.
To rotate a key, a <message/> must be sent to all people which got the shared secret. The <message/> MUST contain a XEP-0373 <openpgp/> element with a <signcrypt/> element whose payload is an encrypted <revoke/> element qualified by the 'urn:xmpp:openpgp:pubsub:0' namespace, which MUST have a 'jid' attribute whose value is the JID of the pubsub/PEP service where the node is, a 'node' attribute whose value is the name of the node, and an 'id' attribute whose value is the shared secret ID.
Optionally, the <revoke/> element MAY have one or more <reason/> child element(s) which contain a human readable message explaining the reason of the revocation. The <reason/> element MAY contain an 'xml:lang' attribute with the language code of the text, and there MAY be several <reason/> elements if and only if they have distinct xml:lang. A revoked key MUST NOT be used to encrypt new items.
Then a new shared secret MUST be generated and transmitted to all participants (excluding those who have seen their access revoked) as explained in Transmit Shared Secret.
Note that if an entity's access is revoked, it SHOULD also be removed from the node's whitelist

(if "whitelist" has been used as access model as recommended).
Following example only show the <revoke/> element, as it is normally put as an encrypted payload of OpenPGP for XMPP (XEP-0373) [9] <signcrypt/> payload.

Listing 3: revoke element (normally encrypted as part of XEP-0373's signcrypt payload)

```
<revoke xmlns='urn:xmpp:openpgp:pubsub:0' jid='pubsub.capulet.lit'
    node='123abc' id='1234-abcd-5678-efgh'>
     <reason>Revoked access from an entity</reason>
  </revoke>
```

Items published before the key rotation SHOULD NOT be re-encrypted as it would be resource intensive, and revoked entities may have made a copy anyway.
When access to the shared secret is granted to a new entity, all previously used keys SHOULD be transmitted (with their 'revoked' attribute set to "true" as explained in Transmit Shared Secret.) along with the currently used shared secret. This allows the new entity to decrypt the items encrypted with the old keys. The <shared-secret/> elements can either be sent in the same encrypted payload of a single <message/>, or split into multiple encrypted <message/>, it's up to the implementation to decide which is better (keep in mind that too many shared secrets in a single message may reach the maximum stanza size limit at some point, even if this limit is not likely to be reached for most usual cases).

6 Business Rules

The shared secret and revocation SHOULD be generated by node owner.
<message/> with encrypted <shared-secret/> or <revoke/> elements MUST be sent to everybody who must have access to the node. At least one of the encrypted payload SHOULD be encrypted for the sender themselves (either by sending the <message/> directly to the sender, or to decrypt a copy received with Message Carbons (XEP-0280) [10]), so other devices of the sender can get the shared secret too.
When Pubsub Attachments (XEP-0470) [11] are used, the attachment node SHOULD have the same access model (i.e. whitelist as recommended in Security Considerations) as the encrypted node, with authorized entities synchronized (this should be done automatically for fully compliant services). The items published to attachment node itself MUST be encrypted using this protocol. Due to validity check of attachment items, the encrypted element MUST be a child of a XEP-0470's <attachments/> element, instead of being the <item/> payload as usual. Note that this will prevent the summary feature to work with encrypted elements, and the end-user client will have to do the summary itself, this is an inevitable trade-off when using e2ee.
The "pubsub#type" of an encrypted item is always "urn:xmpp:openpgp:pubsub:0", thus no

[9] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.
[10] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[11] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.

information is leaked on the content of the node, and all encrypted nodes can easily be retrieved by using PubSub Type Filtering (XEP-0462) [12].
Note that encrypted items MAY be mixed with plain text items: for instance if a blog is public, but some of its items are private. However the proper handling of this use case is out of scope of this specification.

7 Implementation Notes

This specification uses OpenPGP for XMPP (XEP-0373) [13] independently of OpenPGP for XMPP Instant Messaging (XEP-0374) [14], implementations SHOULD avoid double encryption (e.g. by excluding <openpgp/> element qualified by 'urn:xmpp:openpgp:0' namespace from encryption by XEP-0374 implementation), and MUST ensure that element is correctly decrypted. Double encryption should be checked, and the <openpgp/> element MUST be decrypted even if OXIM (XEP-0374) is not being used.

8 Discovering Support

If a client supports the protocol specified in this XEP, it MUST advertise it by including the "urn:xmpp:openpgp:pubsub:0" discovery feature in response to a Service Discovery (XEP-0030) [15] information request:

Listing 4: Service Discovery information request

```
<iq type='get'
    from='juliet@example.org/balcony'
    to='romeo@example.org/orchard'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 5: Service Discovery information response

```
<iq type='result'
    from='romeo@example.org/orchard'
    to='juliet@example.org/balcony'
    id='disco1'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    ...
    <feature var='urn:xmpp:openpgp:pubsub:0'/>
```

[12] XEP-0462: PubSub Type Filtering <https://xmpp.org/extensions/xep-0462.html>.
[13] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.
[14] XEP-0374: OpenPGP for XMPP Instant Messaging <https://xmpp.org/extensions/xep-0374.html>.
[15] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
    ...
  </query>
</iq>
```

9 Security Considerations

- When receiving a <shared-secret/> or a <revoke/> element, the receiving client MUST ensure that the signing sender is the same as the one for all known shared secrets of this pubsub node (usually a node owner). This prevents a malicious actor for misleading the client into using a non-legitimate secret.
- To limit the surface of attack, the access model of an encrypted node should be set to "whitelist" and only people having the shared key should be allowed to retrieve encrypted items.
- If the shared key is compromised, or a user access is revoked, the key MUST be rotated. However, only new items are encrypted with the new key, any previous item should be considered as compromised too.
- Sometimes client may use metadata to construct item ID, this is notably the case for some Microblogging Over XMPP (XEP-0277) [16] implementation, as the resulting item ID is used to generate user friendly URLs. To avoid metadata leakage, clients SHOULD NOT derivate the item ID from any data of the item when pubsub encryption is used.
- Some protocols define prefixes for nodes, this is for instance the case for XEP-0277 Comments. If the prefix is not essential to make the feature work (which is the case for XEP-0277 comments), the client SHOULD NOT use the usual prefix to avoid leaking informations on the content of the node. The prefix defined in Pubsub Attachments (XEP-0470) [17] MUST be used as usually, as it is essential to retrieve the attachment node, and it's generic and thus doesn't leak information on the node content.
- To decrypt archives and future items, clients most probably cache plain text version of items, and shared secret. If one device with access to an encrypted node is compromised, all items from the node should be considered compromised, and shared secret should be rotated.
- For the same reason, client SHOULD encrypt data at rest (even if the device is stolen, data should not be accessible without some cryptographic key).
- Note that only the shared key sender is authenticated (by OpenPGP for XMPP (XEP-0373) [18]'s <signcrypt/>), not the items publishers, meaning that encrypted items could be published by anybody in possession of the shared secret. Pubsub items authentication will be treated in a separated XEP.

[16] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.
[17] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.
[18] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.

- The shared secret should be long enough to avoid brute force attacks. A secret of at least 32 characters is recommended.

10 IANA Considerations

TODO

11 XMPP Registrar Considerations

TODO

12 XML Schema

TODO

13 Acknowledgements

Thanks to Tim Henkes for his advices and NLNet foundation/NGI0 Discovery for funding.