XEP-0475: Pubsub Signing

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

2022-12-20
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	pubsub-signing

Specifies a mechanism to sign pubsub items

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

There are few ways to authenticate items published via Publish-Subscribe (XEP-0060) [1], and none of them are secure: the publish attribute defined by the pubsub service is not mandatory and can be spoofed by the service itself, and some XEPs such as Microblogging Over XMPP (XEP-0277) [2] have their own mechanism (like <author/> qualified by "http://www.w3.org/2005/Atom" namespace) that is even easier to spoof.
This specification proposes a framework for attaching cryptographic signatures to pubsub items, allowing strong authentication of item authors. This specification only defines the framework, it is designed to be extended by other specifications to use various cryptographic algorithms.

2 Glossary

- **wrapper element**: element wrapping the item to sign, and containing extra metadata
- **signed data**: normalized and serialized wrapped element
- **signing profile**: a specialisation of this specification for a specific cryptographic algorithm.
- **signature**: element containing the signature itself (which is detached from the signed data).
- **C14N**: Canonical XML (version 2.0 is used in this specification), a way to normalize XMP to have the same data to sign.

3 Requirements

The design goals of this XEP are:

- it must be possible to sign plain text items as well as end-to-end encrypted items;
- it must be backwards compatible: attaching a signature must work with existing specifications so that clients that do not support pubsub signatures can continue to work as usual;
- it must be possible to sign an item by several authors;
- it should be possible to have different signers from the item publisher;

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[2] XEP-0277: Microblogging over XMPP <https://xmpp.org/extensions/xep-0277.html>.

4 Overview

To sign a pubsub item, the signature and the signed data are separated. Signed data is a wrapper element comprising essential data such as signers, and the item to be signed. The wrapper element is then normalized, serialized and signed. The signature and additional data of the wrapper element are then publised as Pubsub Attachments (XEP-0470) [3]. In case of multiple signers, each signer publish their own signature as an attachment.
To verify a signature, the process is similiar: the receiving client builds the same wrapper element, normalize and serialize it, and uses it to validate the given signature(s).

5 Signing a Pubsub Item

To sign a pubsub item, a <sign-data/> wrapper element qualified by the 'urn:xmpp:pubsub-signature:0' namespace is created. This element MUST contain at least one 'to' element which MUST have a 'jid' attribute whose value is the intended recipient's XMPP address. The XMPP address found in the 'to' element's 'jid' attribute SHOULD be without Resourcepart (i.e., a bare JID).
The <sign-data/> element MUST contain exactly one <time/> element. The <time/> element MUST have a 'stamp' attribute which contains the timestamp of the moment when the element is being signed in the DateTime format as specified in XMPP Date and Time Profiles (XEP-0082) [4]
The <sign-data/> element MUST contain one or more <signer/> element(s) which MUST possess a 'jid' attribute whose value is the bare JID of the signer.
One or more external elements specified by signing profile MAY be added
The item to sign MUST be added as a child of the <sign-data/> element. If the wrapped <item/> element possesses a 'publisher' attribute, it MUST be removed when added to the wrapper element. As item ID can be added or modified by the Pubsub/PEP service, if the <item/> has an 'id' attribute, it MUST be removed too when added to the wrapper element, thus the item ID is not part of the signed data.
Then the resulting item is put to canonical form by applying C14N v2.0 specification.
The resulting element in canonical form is then serialized and signed.
Below is an example of wrapper element:

Listing 1: Wrapper Element (Before Normalization)

```
<sign-data>
      <to jid='juliet@capulet.lit' />
      <time stamp='2022-10-16T18:39:03Z' />
      <signer>juliet@capulet.lit</signer>
      <item>
        <entry xmlns='http://www.w3.org/2005/Atom'>
```

[3] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.
[4] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

```
          <author>
            <name>Juliet Capulet</name>
            <uri>xmpp:juliet@capulet.lit</uri>
          </author>
          <title type='text'>She is so pretty! </title>
          <published>2022-10-16T18:39:02Z</published>
        </entry>
      </item>
    </sign-data>
```

The normalized form is as follow:

Listing 2: Wrapper Element (After Normalization)

```
<sign-data><to jid="juliet@capulet.lit"></to><time stamp="2022-10-16
    T18:39:03Z"></time><signer>juliet@capulet.lit</signer><item><entry
     xmlns="http://www.w3.org/2005/Atom"><author><name>Juliet Capulet<
    /name><uri>xmpp:juliet@capulet.lit</uri></author><title type="text
    ">She is so pretty!</title><published>2022-10-16T18:39:02Z</
    published></entry></item></sign-data>
```

The signature is then put as an Pubsub Attachments (XEP-0470) [5]. The attachment is a <signature/> element qualified by the 'urn:xmpp:pubsub-signing:0' namespace. The attachment MUST contain the same <time/> and <signer/> elements in the same order as in the <sign-data/> element. If any signing profile extra elements have been used in <sign-data/>, they MUST be added too in the same order as in <sign-data/>. Then the signature is added in an element specified in the signing profile specification.
Each signer entity MUST publish a <signature/> attachment signed with their own encryption keys.
If the pubsub item is encrypted, the signature MUST be done on the plain text version of the item **before** the encryption of the item. The <signature/> attachment SHOULD be encrypted too.

Listing 3: Juliet Publish Her Signature as an Attachment

```
<iq from='juliet@capulut.lit/chamber'
    id='signature_1'
    to='juliet@capulet.lit'
    type='set'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:pubsub-attachments:1/xmpp:juliet@capulet.
       lit?;node=urn%3Axmpp%3Amicroblog%3A0;item=random-thoughts-12bd
       '>
      <item id='juliet@capulet.lit'>
        <attachments>
```

[5] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.

```
          <signature xmlns='urn:xmpp:pubsub-signing:0' timestamp="
              2022-10-16T18:39:04Z">
            <time stamp='2022-10-16T18:39:03Z' />
            <signer>juliet@capulet.lit</signer>
            <example-signature xmlns='https://example.org/signature'>
              <!-{}- SOME PAYLOAD -{}->
            </example-signature>
          </signature>
        </attachments>
      </item>
    </publish>
  </pubsub>
</iq>
```

5.1 rationales

The reason we use Pubsub Attachments (XEP-0470) [6] instead of directly signing the target item is that we need to be backwards compatible, so we cannot replace the target item with another element, nor is it possible to add a sibling element to item's payload (this would not be compliant with Publish-Subscribe (XEP-0060) [7] specification). This requires detaching the signature from the <item/> element itself, and Pubsub Attachments (XEP-0470) [8] are dedicated to attaching data to items, so a viable solution.

5.2 Summarizing

To summarize signatures as explained in Pubsub Attachments (XEP-0470) [9] the <signer/> elements are grouped into a <signature/> element qualified by the 'urn:xmpp:pubsub-signing:0' namespace. This allows a client to easily know that an item is signed, and to obtain the IDs of attachments that need to be retrieved to validate signatures.

Listing 4: Juliet Get Summary of Signatures of an Item

```
<iq from='juliet@capulet.lit'
    id='summary_123'
    to='juliet@capulet.lit/chamber'
    type='result'>
    <pubsub xmlns='http://jabber.org/protocol/pubsub'>
      <items node='urn:xmpp:pubsub-attachments:summary:1/
          urn:xmpp:microblog:0'>
        <item id='some-post-with-several-signatures-0adf'>
          <summary xmlns='urn:xmpp:pubsub-attachments:summary:1'>
```

[6] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.
[7] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[8] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.
[9] XEP-0470: Pubsub Attachments <https://xmpp.org/extensions/xep-0470.html>.

```
            <!-{}- … -{}->

            <signature xmlns='urn:xmpp:pubsub-signing:0' >
              <signer>juliet@capulet.lit</signer>
              <signer>romemo@montague.lit</signer>
            </signature>

            <!-{}- … -{}->
          </summary>
        </item>
      </items>
    </pubsub>
  </iq>
```

5.3 Signature Validation

Once one or more signatures have been found in an item attachment, a client SHOULD validate them. To do this, it builds a wrapper element with the target item as explained in Signing a Pubsub Item, and validate it with each signature. Validation mechanism depends of the signing profile.

6 Business Rules

C14N 2.0 defines parameters for the algorithm. For this specification, default values MUST be used, i.e. *IgnoreComments* is true, *TrimTextNodes* is true, *PrefixRewrite* is none, and *QNameAware* is the empty set. In other terms: there must be no comments, text nodes must be trimmed, prefixes are left unchanged, and no nodes must be processed as QName-valued.
Once the signature has been validated, it's the original item which MUST be used, as usual, not the normalized form. The original item has attributes missing from the normalized form ('published' and 'id' attribute), and spaces are trimmed, but they may be significant (e.g. in a dataform <value/>).
It is essential to use the same <to/>, <time/>, <signer/> and signing profile extra elements in the <signature/> element put in attachment and in wrapper <sign-data/> element used for signed data, as it is necessary for receiving client to re-build the wrapper element and then validate the signature.

7 Implementation Notes

The client validating signatures should display a message or indicator depending on the validation result:

- If one of the signatures doesn't validate, the client SHOULD display a prominent warning message explicitely stating that the signature is not validated and that the message is probably spoofed.
- If the signature is validated but at least one of the signers's fingerprints is not trusted, the client SHOULD display a warning message stating that the signature is validated but unreliable, and that the message may be forged.
- If all signatures are validated **and** all signers' fingerprints are trusted, the client SHOULD display an information message or indication that the item is signed by one or more trusted signers.

8 Discovering Support

If a client supports the protocol specified in this XEP, it MUST advertise it by including the "urn:xmpp:pubsub-signing:0" discovery feature in response to a Service Discovery (XEP-0030) [10] information request:

Listing 5: Service Discovery information request

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

Listing 6: Service Discovery information response

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

9 Security Considerations

Signature is intimely linked to the trust in the fingerprint of the encryption keys. A warning SHOULD be displayed by a client if a signature is valid but the signing entity's fingerprints are

[10] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

not trusted. Trust should be done through an external channel (outside of XMPP), preferably face-to-face.
Security considerations of the signing profile applies.

10 IANA Considerations

TODO

11 XMPP Registrar Considerations

TODO

12 XML Schema

TODO

13 Acknowledgements

Thanks to NLnet foundation/NGI0 Discovery for funding.