XEP-0510: End-to-End Encrypted Contacts Metadata

Jérôme Poisson
mailto:goffi@goffi.org
xmpp:goffi@jabber.fr
https://www.goffi.org

2026-02-10
Version 0.1.0

Status	Type	Short Name
Experimental	Standards Track	contacts

This specification describes how to encrypt contacts metadata to minimize server exposure.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Roster is a central component of the XMPP specifications, serving as the mechanism for indicating to the server and other devices who we know, along with associated metadata and presence permission data. While convenient, this information carries significant privacy implications. In the context of ongoing efforts to reduce metadata exposure in XMPP, this document specifies a method for managing contact data in a more privacy-respectful way by using end-to-end encryption, while retaining the original roster mechanism described in RFC 6121 [1] for server-required features and backward compatibility.

2 Requirements

The design goals of this XEP are:

- Reduce metadata exposure: no contact data should be exposed to server beyond what is required for presence permission and backward compatibility.
- Mandatory end-to-end encryption.
- Contact data must not be tied to a JID or any domain-specific identifier.
- The mechanism must be usable in serverless configurations, without relying on server-side storage or processing of contact data.

3 Design Decisions

A few words to explain the design decisions. There are basically two ways to encrypt contact metadata:

1. By using the existing roster mechanism, adding a new element to handle metadata, and implementing end-to-end encryption for this element.
2. By implementing another metadata mechanism, exploiting other XMPP mechanisms. Publish-Subscribe (XEP-0060) [2] is the most suitable one for this use case.

Option 2 has been chosen for the following reasons:

[1] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

- The <item/> element of the roster mechanism, as specified in RFC 6121 [3] §2.1.2, has a mandatory "jid" element, and as per requirements we don't want to tie contact data to a JID.
- The roster's <item/> element has not been explicitly designed to accept extra elements, and most implementations probably do not support them. While this could be added, there is a risk that some implementations remove unknown elements, and it would take years for the entire ecosystem to support them. Even then, there will always be a risk that an old client removes unknown metadata, as experience has shown with legacy bookmark mechanisms (with Bookmark Storage (XEP-0048) [4] clients could remove unknown elements).
- There is no end-to-end encryption mechanism for roster. While this could be added, it would certainly bring compatibility issues with existing implementations, would still leave the JID visible in clear text (as it is mandatory on the <item/> element), and would require additional extensions. On the other hand, there are already end-to-end encryption mechanisms for Publish-Subscribe (XEP-0060) [5] (notably OpenPGP for XMPP Pubsub (XEP-0473) [6]).

4 Service and Well-Known Nodes

Encrypted contacts use two well-known nodes:

- **urn:xmpp:contacts** where contact information is stored.
- **urn:xmpp:contacts-groups** where groups used to classify contacts are stored.

These nodes MUST be set on the Personal Eventing Protocol (XEP-0163) [7] service (except in serverless configurations, where they MAY be set on another relevant pubsub service, see Business Rules), and MUST follow Best Practices for Persistent Storage of Private Data via Publish-Subscribe (XEP-0223) [8]. The IDs used for the pubsub items MUST NOT have any semantic meaning, and in particular MUST NOT be derived from any contact-related data.
Both node items MUST be end-to-end encrypted. The currently recommended method is using OpenPGP for XMPP Pubsub (XEP-0473) [9] as OX is well adapted for this use case. However, another encryption method MAY be used in the future if proven better.

[3] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[4] XEP-0048: Bookmark Storage <https://xmpp.org/extensions/xep-0048.html>.
[5] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[6] XEP-0473: OpenPGP for XMPP Pubsub <https://xmpp.org/extensions/xep-0473.html>.
[7] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[8] XEP-0223: Best Practices for Persistent Storage of Private Data via Publish-Subscribe <https://xmpp.org/extensions/xep-0223.html>.
[9] XEP-0473: OpenPGP for XMPP Pubsub <https://xmpp.org/extensions/xep-0473.html>.

4.1 Contacts Node

Contact metadata are stored on the 'urn:xmpp:contacts' node. Each published item MUST have either a <contact/> element qualified by the 'urn:xmpp:contacts:0' namespace or a <reserved/> element qualitifed by the 'urn:xmpp:contacts:0' namespace (see below).
The <contact/> element MAY have a 'name' attribute to specify the "handle" to be associated with the contact, in the same way as for roster items as described in RFC 6121 [10] §2.1.2.4.
The <contact/> element may have the following child elements:

4.1.1 identity

The <contact/> MUST have a way to identify the contact, by having one or more <identity/> elements qualified by the 'urn:xmpp:contacts:0' namespace. This element MUST have a 'type' attribute. The 'type' attribute can have the value "jid" when the contact is identified by its bare JID; in that case, its content is the bare JID of the contact. Future specifications may propose other <identity/> types to identify contacts by other means (e.g., cryptographic fingerprint).

4.1.2 group

In a similar way as for RFC 6121 [11] roster, zero, one, or more groups may be associated with a contact. However, groups use IDs and are defined in the separate node 'urn:xmpp:contacts-groups'. This allows group names or other metadata to be modified without having to change all references to this group.
A group is specified by the <group/> element, which MUST have an 'id' attribute set to the ID of the desired group. This ID is associated with group metadata in the 'urn:xmpp:contacts-groups' node.

4.1.3 Contacts Example

Romeo adds Juliet to his contacts:

Listing 1: Setting a Contact Item

```
<iq type='set'
  from='romeo@example.net/orchard'
  to='romeo@example.net'
  id='contact1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
```

[10] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[11] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.

```
    <publish node='urn:xmpp:contacts'>
      <item id='a1b2c3d4'>
        <contact xmlns='urn:xmpp:contacts:0' name='Juliet␣Capulet'>
          <identity type='jid'>juliet@example.org</identity>
          <group id='grp-789'/>
          <group id='grp-123'/>
        </contact>
      </item>
    </publish>
  </pubsub>
</iq>
```

Note: the example is in clear here, but in real use cases it is end-to-end encrypted.

4.2 Groups Node

Group metadata are stored on the 'urn:xmpp:contacts-groups' node. Each published item MUST have either a <group/> element qualified by the 'urn:xmpp:contacts:0' namespace or a <reserved/> element qualitifed by the 'urn:xmpp:contacts:0' namespace (see below).
The <group/> element MUST have an 'id' attribute set to a unique value. This ID defines the group identity and MUST NOT change as long as the group is not deleted.
The <group/> element MUST have a 'name' attribute to specify the "handle" to be associated with the group. This name MAY be changed in an existing group.
The <group/> element may have the following child element:

4.2.1 description

An optional <description/> element may be used to describe the group.

4.2.2 Groups Example

Romeo adds the groups where he places Juliet:

Listing 2: Setting Group Items

```
<iq type='set'
  from='romeo@example.net/orchard'
  to='romeo@example.net'
  id='group1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:contacts-groups'>
      <item id='grp-789'>
        <group xmlns='urn:xmpp:contacts:0' id='grp-789' name='Friends'
          />
```

```
      </item>
      <item id='grp-123'>
        <group xmlns='urn:xmpp:contacts:0' id='grp-123' name='House␣of
            ␣Capulet'/>
      </item>
    </publish>
  </pubsub>
</iq>
```

Note: the example is in clear here, but in real use cases it is end-to-end encrypted.

4.3 Reserved Element

Since contact and group metadata are stored one per pubsub item, the number of items may reveal the size of the contact list. To mitigate this, clients MAY publish additional items without actual data, which should be ignored by other clients. This prevents the server from precisely determining the number of contacts or groups by simply counting encrypted pubsub items.
For either the contacts or groups node, a client MAY publish an item containing only a <reserved/> element qualified by the 'urn:xmpp:contacts:0' namespace. A client may publish as many such items as desired, replace an existing <contact/> or <group/> element with a reserved one, or vice versa.

4.3.1 Reserved Example

Romeo wants to hide the size of his contact list by publishing reserved items:

Listing 3: Setting Reserved Items

```
<iq type='set'
  from='romeo@example.net/orchard'
  to='romeo@example.net'
  id='reserved1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='urn:xmpp:contacts'>
      <item id='a1b2c3d5'>
        <reserved xmlns='urn:xmpp:contacts:0'/>
      </item>
    </publish>
  </pubsub>
</iq>
```

Note: the example is in clear here, but in real use cases it is end-to-end encrypted.

5 Business Rules

- The RFC 6121 [12] roster mechanism is still used, as it is necessary for presence permission handling and backward compatibility. However, when encrypted contacts metadata are used, the roster should be kept minimal: the 'name' attribute and <group/> element SHOULD NOT be used. Instead, an encrypted contact with the same JID SHOULD be used to store the name and group metadata.
- As random IDs are used, it is possible that two or more <contact/> elements referring to the same contact are published (e.g., due to a race condition between two clients updating the same contact). In this case, the receiving client MUST merge the data from both elements (overwriting data from the previous item in case of conflict, in the order of item reception). The receiving client MAY overwrite the last seen <contact/> element with the result of the merge, and MAY either delete the older items or replace them with <reserved/> elements.
- In serverless setups, as there is no server to offer a PEP service, the well-known node may be placed on whichever service is used to replace Personal Eventing Protocol (XEP-0163) [13].

6 Discovering Support

If a client supports encrypted contacts, it MUST advertise it by including the "urn:xmpp:contacts:0" discovery feature in response to a Service Discovery (XEP-0030) [14] information request:

Listing 4: Service Discovery information request

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

Listing 5: Service Discovery information response

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

[12] RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence <http://tools.ietf.org/html/rfc6121>.
[13] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.
[14] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

```
    ...
    <feature var='urn:xmpp:contacts:0'/>
    ...
  </query>
</iq>
```

7 Security Considerations

The security consideration of used XEPs apply here and must be carrefully checked, in paritcular the ones from Publish-Subscribe (XEP-0060) [15] Best Practices for Persistent Storage of Private Data via Publish-Subscribe (XEP-0223) [16] and OpenPGP for XMPP Pubsub (XEP-0473) [17].

8 IANA Considerations

This document does not require interaction with the Internet Assigned Numbers Authority (IANA) [18].

9 XMPP Registrar Considerations

TODO

10 Acknowledgements

This work has been done for the Serverless and Metadata Reduction for XMPP project. Many thanks to NLNet foundation and NGI Zero Core for funding the work on this specification.

11 XML Schema

TODO

[15] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.
[16] XEP-0223: Best Practices for Persistent Storage of Private Data via Publish-Subscribe <https://xmpp.org/extensions/xep-0223.html>.
[17] XEP-0473: OpenPGP for XMPP Pubsub <https://xmpp.org/extensions/xep-0473.html>.
[18] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.