XEP-0434: Trust Messages (TM)

Melvin Keskin
mailto:melvo@olomono.de
xmpp:melvo@olomono.de

2021-10-04
Version 0.6.0

Status	Type	Short Name
Experimental	Standards Track	TM

This document specifies a way to communicate the trust in public long-term keys used by end-to-end encryption protocols from one endpoint to another.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

End-to-end encryption without verifying the authenticity of the exchanged public long-term keys only enables the endpoints to protect their communication against passive attacks. This means an attacker cannot read encrypted messages in transit without actively intervening in the key exchange. However, without any other precautions, active attacks are still possible. If an attacker replaces the exchanged keys with malicious ones or introduces an additional malicious endpoint, the end-to-end encrypted messages can be read and manipulated by the attacker.
When using end-to-end encryption where public long-term keys are transmitted over a channel that is not protected against active attacks, the authenticity of those keys is not guaranteed. Such a key has to be authenticated by the receiving endpoint over another channel that is already protected against active attacks to maintain the confidentiality of sent messages and ensure the authenticity and integrity of received messages. Trust messages can be used to transfer the needed data via XMPP for performing such an authentication. Furthermore, they can transmit the data used for distrusting a key.

2 Glossary

Endpoint Communication endpoint owning exactly one public long-term key. In most cases that is an XMPP client instance. In the terminology of OMEMO Encryption (XEP-0384) XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>., that is a device. To cover also the possibility for using multiple endpoints on the same physical device and via the same client instance, the general term endpoint is used.

Key authentication Verification that a key received over an insecure channel is actually the one of the assumed endpoint

Key identifier Identifier of a key (e.g., a hash value / fingerprint or the key itself)

Public key certificate Electronic document used to prove the authenticity of a public key

Trust message XMPP message that indicates whether specific keys are trusted or untrusted by the sending endpoint. A trust message for an endpoint's key contains the key identifier of the given key.

3 Why Trust Messages?

Trust messages can be used in conjunction with an end-to-end encryption protocol such as OpenPGP for XMPP (XEP-0373) [1] or OMEMO Encryption (XEP-0384) [2] to automatically or semi-automatically establish secure channels protected against active attacks. This protocol

[1] XEP-0373: OpenPGP for XMPP <https://xmpp.org/extensions/xep-0373.html>.
[2] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.

specifies how trust messages are transmitted and protocols such as Automatic Trust Management (ATM) (XEP-0450) [3] specify how and for which purpose they are processed.

3.1 General Advantages

Trust messages have the following advantages:

3.1.1 No Permanent Storage

A permanent storage is not needed. Cached data can be be removed as soon as the recipient made use of them. E.g., a server-side storage holding available data because it does not know when a client needs to access them, as for certificate-based approaches, is unnecessary.

3.1.2 No Additional Infrastructure

An infrastructure in additional to the one used for messages is not needed because trust messages are ordinary messages. E.g., a server-side storage and a mechanism for accessing it other than those used for messages is unnecessary.

3.2 Advantages with Encryption

The authenticity and integrity of trust messages are ensured by a signing mechanism. If trust messages are additionally encrypted, they have the following advantages:

3.2.1 Same Cryptographic Properties

The cryptographic properties of the encryption protocol are applied to the trust messages. Properties such as confidentiality, forward secrecy and deniability can have a positive impact on the authentication of keys. Hence, trust messages, in contrast to certificates, can e.g. be deniable toward a third party if the encryption protocol provides that kind of deniability.

3.2.2 Confidential Communication of Trust

The fact that an endpoint trusts a key or not can be kept confidential toward an attacker by encrypting those messages and sending them only to endpoints with authenticated keys. This means that an attacker cannot detect by the content of a trust message whether a specific key is trusted by the sender. If the trust message is sent for an authentication of a key, the involved parties of that authentication will therefore stay anonymous toward an attacker.

[3] XEP-0450: Automatic Trust Management (ATM) <https://xmpp.org/extensions/xep-0450.html>.

The encryption protects against passive attacks on the transmission of the trust messages. That way, an attacker cannot read the content of the trust messages.
The restriction to send trust messages only to endpoints with authenticated keys in addition to the encryption protects against active attacks on the transmission of the trust messages. An attacker will even after introducing a malicious key not receive a trust message encrypted with that key.

3.2.3 No Selective Blocking

Blocking data used to trust or distrust keys in transit is made more difficult. In particular, an attacker is not able to selectively block transmitted data used to distrust the attacker's key.
If an attacker cannot distinguish whether the data sent from a client is used for trusting or distrusting a key, the attacker can only randomly block some messages or the whole communication. If the communication is already compromised by an active attack, the attacker does not want to stop the whole communication. During that state, the attacker has the possibility to keep on e.g. eavesdropping or altering messages. Therefore, the attacker wants to block data that can lead to excluding the attacker. But the attacker does not want to block the communication itself.
Data that is used by the recipient to distrust the attacker's key would make it impossible for the attacker to continue to encroach on the communication. Thus, it is important to prevent an attacker from blocking data used for making trust decisions. E.g., an approach using certificates permanently stored on a server cannot prevent an attacker from specifically blocking such data because certificates have to be discoverable and identifiable as such.
It might be possible to distinguish an encrypted trust message from other encrypted messages by analyzing the network traffic over a period of time. However, the mitigation of that issue is out of scope.

4 Trust Message Structure

A trust message (i.e., the root <message/> element)

- MUST contain exactly one <trust-message/> direct child element that
  - MUST be signed in a way to ensure its authenticity and integrity.
  - SHOULD be encrypted to ensure its confidentiality.
  - MUST have an *xmlns* attribute specifying its namespace *urn:xmpp:tm:1*.
  - MUST have a *usage* attribute specifying the namespace of the protocol that uses the trust message for a specific purpose.
  - MUST have an *encryption* attribute specifying the namespace of the encryption protocol that uses the keys.

  - MUST contain at least one <key-owner/> direct child element that
    * MUST have a *JID* attribute specifying the bare JID of the key owner.
    * MUST contain at least one <trust/> or <distrust/> direct child element indicating the trust respectively distrust in a key. Each <trust/> and <distrust/> element MUST contain exactly one Base64-encoded (see RFC 4648 [4]) key identifier. The kind of identifier that the key's encryption protocol specifies MUST be used.

- SHOULD have a type='chat' attribute which is needed to deliver the trust message to all endpoints (see XEP-0280: Message Carbons).
- SHOULD contain a <store xmlns='urn:xmpp:hints'/> direct child element which is needed to deliver the trust message to each offline endpoint after it went online (see XEP-0313: Message Archive Management and XEP-0334: Message Processing Hints).

The last two points are needed to achieve their mentioned goals because a trust message does not contain a <body> element which would automatically lead to the desired result.
In the following example, two OMEMO Encryption (XEP-0384) [5] keys of Alice are indicated as trusted, one key of Bob is indicated as trusted and two other ones of Bob are indicated as untrusted.

Listing 1: Trust Message Element for Alice's and Bob's OMEMO Keys used by ATM

```
<trust-message xmlns='urn:xmpp:tm:1' usage='urn:xmpp:atm:1' encryption
    ='urn:xmpp:omemo:2'>
  <key-owner jid='alice@example.org'>
    <trust>aFABnX7Q/rbTgjBySYzrT2FsYCVYb49mbca5yB734KQ=</trust>
    <trust>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</trust>
  </key-owner>
  <key-owner jid='bob@example.com'>
    <trust>YjVI04NcbTPvXLaA95RO84HPcSvyOgEZ2r5cTyUs0C8=</trust>
    <distrust>tCP1CI3pqSTVGzFYFyPYUMfMZ9Ck/msmfD0wH/VtJBM=</distrust>
    <distrust>2fhJtrgoMJxfLI3084/YkYh9paqiSiLFDVL2m0qAgX4=</distrust>
  </key-owner>
</trust-message>
```

5 Use Cases

Trust messages MAY be sent unencrypted to solely communicate the trust in specific keys without any other protection. But as described before, the strength of trust messages is the possibility to encrypt their content and choose to which endpoints they are sent according to the trust in the endpoints' keys.

[4] RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.
[5] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.

5.1 Unencrypted Trust Message

The <trust-message/> element MUST be signed before sending for ensuring the authenticity and integrity.

5.2 Encrypted Trust Message

The <trust-message/> element SHOULD be signed and encrypted before sending for applying all advantages the encryption provides. That protects against passive attacks on the transmission of the trust message. The trust message SHOULD only be sent to endpoints whose keys have already been authenticated for also preventing active attacks on the transmission of the trust message.
Describing how the <trust-message/> element has to be used by each existing encryption protocol is out of scope. Stanza Content Encryption (XEP-0420) [6] specifies a common method for encrypting arbitrary elements which can be used by different encryption protocols. When using an encryption protocol such as OMEMO Encryption (XEP-0384) [7] that uses Stanza Content Encryption (XEP-0420) [8] (SCE), the SCE <content/> element MUST contain the <trust-message/> element as a direct child.

5.2.1 SCE Profile

A trust message SCE <envelope/> element

- MUST contain an <rpad/> affix element. This prevents an attacker from finding out the trust message's content by its length and distinguishing a trust message from another kind of SCE message.
- MUST contain a <time/> affix element. This prevents an attacker from delivering trust messages in the wrong order or delivering a former trust message again. If the trust messages are used by the recipient for making trust decisions, it protects the recipient from setting the opposite of the intended trust state by applying trust changes in the wrong order or reapplying a former change.
- SHOULD contain a <from/> affix element. This prevents an attacker from spoofing the sender.
- SHOULD contain a <to/> affix element. This prevents an attacker from spoofing the recipient.

[6] XEP-0420: Stanza Content Encryption <https://xmpp.org/extensions/xep-0420.html>.
[7] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[8] XEP-0420: Stanza Content Encryption <https://xmpp.org/extensions/xep-0420.html>.

Listing 2: SCE Envelope Element Used for a Trust Message from Alice's Endpoint to Carol

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>QHqW2arWFewoERL1a43wonBKpTmsrBWnc1d66HSDq85NgMLmjrDJV9lV</rpad
     >
  <time stamp='2020-01-01T00:00:00'/>
  <from jid='alice@example.org/notebook'/>
  <to jid='carol@example.com'/>
  <content>
    <trust-message xmlns='urn:xmpp:tm:1' usage='urn:xmpp:atm:1'
       encryption='urn:xmpp:omemo:2'>
      <key-owner jid='alice@example.org'>
        <trust>aFABnX7Q/rbTgjBySYzrT2FsYCVYb49mbca5yB734KQ=</trust>
        <trust>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</trust>
      </key-owner>
      <key-owner jid='bob@example.com'>
        <trust>YjVI04NcbTPvXLaA95RO84HPcSvyOgEZ2r5cTyUs0C8=</trust>
        <distrust>tCP1CI3pqSTVGzFYFyPYUMfMZ9Ck/msmfD0wH/VtJBM=</
           distrust>
        <distrust>2fhJtrgoMJxfLI3084/YkYh9paqiSiLFDVL2m0qAgX4=</
           distrust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

6 Implementation Notes

Message Carbons (XEP-0280) [9] minimizes the number of trust messages to be sent while having the same payload because trust messages with the same payload do not have to be sent for each endpoint. In combination with the usage of Message Archive Management (XEP-0313) [10], the delivery of trust messages to temporarily offline endpoints is ensured even if they are available under a different resource after going online than the last known one before going offline.

This specification uses Message Carbons (XEP-0280) [11] for sending a trust message to all endpoints of a contact or to all own endpoints at once. By sending a trust message to the contact, each endpoint of the contact and each own endpoint receives the same trust message by the server. Thus, a client needs to send the same trust message only once.

If not all endpoints of the contact should receive the trust message, the trust message MAY be sent to specific endpoints of the contact but for all own endpoints Message Carbons (XEP-0280) [12] MAY be used and vice versa. Even when a client does not yet have a contact,

[9] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[10] XEP-0313: Message Archive Management <https://xmpp.org/extensions/xep-0313.html>.
[11] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[12] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.

the client MAY use Message Carbons (XEP-0280) [13] for delivering a trust message to all own endpoints by sending it to the own bare JID. If then a client receives a trust message with its own full JID as the sender, it MAY discard that message directly without parsing the content.
Example: Alice's endpoint A1 authenticates the key of her endpoint A2. A1 sends the trust message for A2's key only once to all of Alice's and Bob's endpoints by using Message Carbons (XEP-0280) [14].
Attention: In that context, sending an encrypted trust message to all endpoints of a contact or to all own endpoints does not mean to encrypt it with the keys of all those endpoints. Instead, it only means that all of those endpoints should receive the trust message even if it is not encrypted for some of them and thereby not decryptable by those endpoints. Keep in mind that a trust message SHOULD only be encrypted for endpoints with authenticated keys.

7 Security Considerations

Protocols using trust messages SHOULD specify rules for processing them in order to create or sustain a secure communication. Therefore, those protocols SHOULD state in which cases from which senders trust messages are used for making trust decisions and for which keys they are sent to whom.

8 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA).

9 XMPP Registrar Considerations

9.1 URI Query Types

As authorized by XMPP URI Query Components (XEP-0147) [15], the XMPP Registrar maintains a registry of queries and key-value pairs for use in XMPP URIs (see <https://xmpp.org/registrar/querytypes.html>).

9.1.1 trust-message

An XMPP URI with the *trust-message* query type (defined as *Trust Message URI*) MAY be used to provide a trust message for various purposes and a single key owner out-of-band. Such a URI MAY be encoded as a QR code and used if only a QR code scan is available as a trusted channel.

[13] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[14] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[15] XEP-0147: XMPP URI Query Components <https://xmpp.org/extensions/xep-0147.html>.

E.g., the *initial authentication* needed by Automatic Trust Management (ATM) (XEP-0450) [16] can be performed by scanning a QR code that encodes a Trust Message URI.
Only a Trust Message URI from a trusted source SHOULD be processed because of its impact on the communication's security. Therefore, users SHOULD be asked for confirmation if a Trust Message URI is used to make a trust decision.
The *JID* attribute of the <key-owner/> element MUST be used as the Trust Message URI's path. The first key-value pair of the URI's query MUST represent the *encryption* attribute of the <trust-message/> element. All remaining key-value pairs of the URI's query MUST represent the <trust/> respectively <distrust/> elements of the <key-owner/> element. Each key identifier MUST be Base16-encoded (see RFC 4648 [17]). The key of a key-value pair MUST be the element's respectively attribute's name and the value their content.

Listing 3: Trust Message URI for Bob's OMEMO keys

```
xmpp:bob@example.com?trust-message;encryption=urn:xmpp:omemo:2;trust
    =623548d3835c6d33ef5cb680f7944ef381cf712bf23a0119dabe5c4f252cd02f;
    distrust=
    b423f5088de9a924d51b31581723d850c7cc67d0a4fe6b267c3d301ff56d2413;
    distrust=
    d9f849b6b828309c5f2c8df4f38fd891887da5aaa24a22c50d52f69b4a80817e
```

9.2 Protocol Namespaces

This specification defines the following XMPP namespaces:

- urn:xmpp:tm:1

9.3 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

10 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
           targetNamespace='urn:xmpp:tm:1'
           xmlns='urn:xmpp:tm:1'
```

[16] XEP-0450: Automatic Trust Management (ATM) <https://xmpp.org/extensions/xep-0450.html>.
[17] RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

```
          elementFormDefault='qualified'>

  <xs:element name='trust-message'>
    <xs:complexType>
      <xs:attribute name='usage' type='xs:string' use='required'/>
      <xs:attribute name='encryption' type='xs:string' use='required'/
          >
      <xs:sequence>
        <xs:element ref='key-owner' minOccurs='1' maxOccurs='unbounded
            '/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name='key-owner'>
    <xs:complexType>
      <xs:attribute name='jid' type='xs:string' use='required'/>
      <xs:sequence>
        <xs:element name='trust' type='xs:base64Binary' minOccurs='0'
            maxOccurs='unbounded'/>
        <xs:element name='distrust' type='xs:base64Binary' minOccurs='
            0' maxOccurs='unbounded'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```