XEP-0450: Automatic Trust Management (ATM)

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

2021-10-04
Version 0.4.0

Status	Type	Short Name
Experimental	Standards Track	ATM

This document specifies a way to automatically manage the trust in public long-term keys used by end-to-end encryption protocols.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

End-to-end encryption can easily be used to protect most of the communication nowadays. However, the management of trust in long-term keys used for encryption is still a manual task. With ATM, many steps that can be automated for the trust management are no longer a manual task the users have to take care of.
ATM reduces the steps needed for authenticating or distrusting keys of endpoints between two identities (XMPP accounts / bare JIDs). It uses Trust Messages (XEP-0434) [1] for automatically establishing secure channels protected against active attacks between a new endpoint and existing ones after an initial mutual authentication between the new endpoint and one of the existing ones.
Trust messages are sent encrypted only to endpoints with authenticated keys. ATM preserves the security level as if all endpoints of the two identities authenticated or distrusted their keys manually. It can be used especially for end-to-end encryption protocols such as OMEMO Encryption (XEP-0384) [2] that use one key per endpoint of the same identity.

2 Glossary

External glossary The terms of Trust Messages (XEP-0434) XEP-0434: Trust Messages <https://xmpp.org/extensions/xep-0434.html>.'s glossary are also relevant for this XEP.

Manual key authentication Key authentication with user interaction (e.g., QR code scanning, fingerprint verification)

Automatic key authentication Key authentication without user interaction (e.g., via ATM)

Initial key authentication Key authentication needed for all further automatic key authentications made by ATM

Mutual key authentication Key authentication in which two endpoints authenticate each other's key

Key distrusting Revoking the trust in a key

Manual key distrusting Key distrusting with user interaction (e.g., clicking a "Distrust" button)

Automatic key distrusting Key distrusting without user interaction (e.g., via ATM)

Authentication message Trust message that only contains key identifiers for authentication. If a trust message contains authentication and distrusting parts, the term authentication message is used for referring to the trust message without the distrusting parts.

[1] XEP-0434: Trust Messages <https://xmpp.org/extensions/xep-0434.html>.
[2] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.

Distrust message Trust message that only contains key identifiers for distrusting. If a trust message contains authentication and distrusting parts, the term distrust message is used for referring to the trust message without the authentication parts.

3 Details

The goal of key authentication is to create an end-to-end encrypted communication network exclusively of endpoints with authenticated keys. As a result, every communication channel between those endpoints is resistant against active attacks.
The network of endpoints that authenticated each other's keys can be seen as a complete graph where each endpoint is a node and each mutual authentication is an edge. The number of edges grows for each new endpoint by the number of existing nodes. This is due to the fact that in order to sustain secure channels between all endpoints, a new key has to be authenticated by all n existing endpoints and vice versa.
One of those n mutual authentications requires a third party or user interaction like scanning each other's QR code or comparing each other's key identifier by hand. That is the initial mutual authentication. A trusted third party MAY be used for the initial authentication but a manual authentication SHOULD be preferred since it does not depend on the trust in the third party. The remaining authentications can be automated relying on the secure channel established by the initial mutual authentication and the secure channels already created by the same procedure between the rest of the endpoints.
For creating the described complete graph with n nodes, a total of $T(n) = (n*(n-1))/2 \in O(n^2)$ mutual authentications are needed. When using ATM, only $T(n) = n-1 \in O(n)$ of them have to be made as initial mutual authentications. All remaining authentications can be performed automatically by ATM. Thus, less user interaction is needed for authenticating all keys involved in the secure communication while preserving the same security level as the manual authentication.

4 Use Cases

Everything that is RECOMMENDED by Trust Messages (XEP-0434) [3] MUST be applied. Trust Message URIs as specified by Trust Messages (XEP-0434) [4] SHOULD be used for the initial authentications.
The examples contain OMEMO Encryption (XEP-0384) [5] as the encryption protocol because it uses one key per endpoint of the same identity. Nevertheless, other encryption protocols MAY be used as stated in Trust Messages (XEP-0434) [6].
If the encryption protocol used with ATM involves only one key for all endpoints of the same identity, only the use cases for authenticating and distrusting keys of a contact's endpoint

[3] XEP-0434: Trust Messages <https://xmpp.org/extensions/xep-0434.html>.
[4] XEP-0434: Trust Messages <https://xmpp.org/extensions/xep-0434.html>.
[5] XEP-0384: OMEMO Encryption <https://xmpp.org/extensions/xep-0384.html>.
[6] XEP-0434: Trust Messages <https://xmpp.org/extensions/xep-0434.html>.

are relevant. Those use cases are marked with OOK for *only one key*. Additionally, the trust messages are always sent to own endpoints because they use the same key as the sending endpoint. Therefore, the parts *..., to each (other) own endpoint with an already authenticated key.* and *... as soon as X authenticated Y's key* are not of interest.
Note that the examples in the following use cases are consecutive and therefore must be read chronologically to properly understand them. Since ATM uses Stanza Content Encryption (XEP-0420) [7], only the SCE <envelope/> elements are shown.
Alice would like to use OMEMO when communicating with Bob. Alice has the endpoints A1, A2 and A3. Bob has the endpoint B1. A1 has already authenticated A2's key. The other endpoints have not authenticated each other's key.

4.1 Authenticating the Key of a Contact's Endpoint

4.1.1 Sending

Example: Remember that A1 has already authenticated A2's key. A1 authenticates B1's key.
An endpoint that initially authenticates the key of a contact's endpoint MUST send an authentication message for ...
... the key that has been authenticated, to each own endpoint with an already authenticated key.

Listing 1: A1 sends an authentication message for B1's key to A2

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>QHqW2arWFewoERL1a43wonBKpTmsrBWnc1d66HSDq85NgMLmjrDJV9lV</rpad
      >
  <time stamp='2020-01-01T12:00:00'/>
  <from jid='alice@example.org/A1'/>
  <to jid='alice@example.org'/>
  <content>
    <trust-message xmlns='urn:xmpp:tm:1' usage='urn:xmpp:atm:1'
        encryption='urn:xmpp:omemo:2'>
      <key-owner jid='bob@example.com'>
        <trust>YjVI04NcbTPvXLaA95RO84HPcSvyOgEZ2r5cTyUs0C8=</trust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

... each already authenticated key of all own endpoints, to the endpoint whose key has been authenticated.

Listing 2: A1 sends an authentication message for A2's key to B1

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

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>Wvj25aDkNbAnSxMIDQo1pjIKRowIMGrF72hSJeXS</rpad>
  <time stamp='2020-01-01T12:00:01'/>
  <from jid='alice@example.org/A1'/>
  <to jid='bob@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>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

4.1.2 Receiving

An endpoint that receives an authentication message for a key of a contact's endpoint from ...
... an own endpoint ...
Example: A2 authenticates B1's key by the authentication message from A1 (see Example 1 for the corresponding authentication message) as soon as A2 authenticated A1's key.
... or another endpoint of that contact ...
Example: B1 authenticates A2's key by the authentication message from A1 (see Example 2 for the corresponding authentication message) as soon as B1 authenticated A1's key.
... MUST authenticate the key as soon as the receiving endpoint authenticated the key of the endpoint that sent the authentication message.

4.2 Authenticating the Key of an Own Endpoint

4.2.1 Sending

Example: Remember that A2 has already authenticated A1's and B1's key. A2 authenticates A3's key.
An endpoint that initially authenticates the key of an own endpoint MUST send an authentication message for ...
... the key that has been authenticated to each other endpoint with an already authenticated key.

Listing 3: A2 sends an authentication message for A3's key to B1 and by using Message Carbons also to A1

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>O2vRKkmtsXsKSk2hPDkrpQQ4Og272qFGB1Srp64vaDrMTNhrV6</rpad>
  <time stamp='2020-01-01T14:00:01'/>
  <from jid='alice@example.org/A2'/>
```

```
  <to jid='bob@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>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</trust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

Listing 4: A2 would send an authentication message for A3's key only to A1 if there were no contacts with authenticated keys

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>s5jP95kvpRNg92XLLo8OkLCvUDT53S</rpad>
  <time stamp='2020-01-01T14:00:00'/>
  <from jid='alice@example.org/A2'/>
  <to jid='alice@example.org'/>
  <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>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</trust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

... each already authenticated key of all endpoints to the endpoint whose key has been authenticated.

Listing 5: A2 sends an authentication message for A1's and B1's key to A3

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>98WA6U92twcVkAXM44UU</rpad>
  <time stamp='2020-01-01T14:00:02'/>
  <from jid='alice@example.org/A2'/>
  <to jid='alice@example.org'/>
  <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>883dkfJVAmUkg74v1fqqoA+AhorA1R1+67GwijiS4z0=</trust>
      </key-owner>
      <key-owner jid='bob@example.com'>
        <trust>YjVI04NcbTPvXLaA95RO84HPcSvyOgEZ2r5cTyUs0C8=</trust>
      </key-owner>
    </trust-message>
```

```
  </content>
</envelope>
```

4.2.2 Receiving

An endpoint that receives an authentication message for a key of an own endpoint from another own endpoint MUST authenticate the key as soon as the receiving endpoint authenticated the key of the endpoint that sent the authentication message.
Example: A1 authenticates A3's key by the authentication message from A2 (see Example 3 or Example 4 for the corresponding authentication message) as soon as A1 authenticated A2's key.

4.3 Distrusting the Key of an Own Endpoint

4.3.1 Sending

Example: Remember that A1 has already authenticated A2's, A3's and B1's key. A1 distrusts A3's key.
An endpoint that initially distrusts the key of an own endpoint MUST send a distrust message for that key to each other endpoint with an already authenticated key.

Listing 6: A1 sends a distrust message for A3's key to B1 and by using Message Carbons also to A2

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>NF5MOJdt8TBbItt4AHXOUKWncRmw5B</rpad>
  <time stamp='2020-01-01T16:00:01'/>
  <from jid='alice@example.org/A1'/>
  <to jid='bob@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'>
        <distrust>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</
            distrust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

Listing 7: A1 would send a distrust message for A3's key only to A2 if there were no contacts with authenticated keys

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>798BFSTQqPjVtiLok3EGtQ7VgB3GGP7eT9P4Fh05</rpad>
  <time stamp='2020-01-01T16:00:00'/>
  <from jid='alice@example.org/A1'/>
```

```
  <to jid='alice@example.org'/>
  <content>
    <trust-message xmlns='urn:xmpp:tm:1' usage='urn:xmpp:atm:1'
        encryption='urn:xmpp:omemo:2'>
      <key-owner jid='alice@example.org'>
        <distrust>IhpPjiKLchgrAG5cpSfTvdzPjZ5v6vTOluHEUehkgCA=</
            distrust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

4.3.2 Receiving

An endpoint that receives a distrust message for a key of an own endpoint from another own endpoint MUST distrust the key as soon as the receiving endpoint authenticated the key of the endpoint that sent the distrust message.
Example: A2 distrusts A3's key by the distrust message from A1 (see Example 6 or Example 7 for the corresponding distrust message) as soon as A2 authenticated A1's key.

4.4 Distrusting the Key of a Contact's Endpoint

4.4.1 Sending (OOK)

Example: Remember that A1 has already authenticated A2's and B1's key but distrusted A3's key. A1 distrusts B1's key.
An endpoint that distrusts the key of a contact's endpoint MUST send a distrust message for that key to each other own endpoint with an authenticated key.

Listing 8: A1 sends a distrust message for B1's key to A2

```
<envelope xmlns='urn:xmpp:sce:1'>
  <rpad>x4LJDawLHgnTJRC7T1mndKEQLPR658NQmXAPQRVnhM1QQ861ve</rpad>
  <time stamp='2020-01-01T18:00:00'/>
  <from jid='alice@example.org/A1'/>
  <to jid='alice@example.org'/>
  <content>
    <trust-message xmlns='urn:xmpp:tm:1' usage='urn:xmpp:atm:1'
        encryption='urn:xmpp:omemo:2'>
      <key-owner jid='bob@example.com'>
        <distrust>YjVI04NcbTPvXLaA95RO84HPcSvyOgEZ2r5cTyUs0C8=</
            distrust>
      </key-owner>
    </trust-message>
  </content>
</envelope>
```

4.4.2 Receiving

An endpoint that receives a distrust message for a key of a contact's endpoint from ...
... another endpoint of that contact ...
Example: B1 distrusts A3's key by the distrust message from A1 (see Example 6 for the corresponding distrust message) as soon as B1 authenticated A1's key.
... or an own endpoint ...
Example: A2 distrusts B1's key by the distrust message from A1 (see Example 8) as soon as A2 authenticated A1's key.
... MUST distrust the key as soon as the receiving endpoint authenticated the key of the endpoint that sent the distrust message.

5 Implementation Notes

5.1 Storing Trust Message Information from Endpoints with Unauthenticated Keys

A client MUST save the information of a trust message until the key of the endpoint that sent the trust message is authenticated, so that the key can then be authenticated or distrusted. Storing data of a trust message from an endpoint with an unauthenticated key is necessary because the receiving endpoint can only use that data after authenticating the sending endpoint's key and that data might not be received again. Afterwards the information of the trust message MAY be deleted.
Example: When Alice's endpoint A1 authenticates the key of Bob's endpoint B1, A1 sends a trust message containing the keys of Alice's other endpoint A2 to B1. If B1 has not already authenticated A1's key, B1 stores the information provided by the trust message. B1 authenticates A1's key and is then able to automatically authenticate A2's key.

5.2 Storing Trust Message Information for Unknown Keys

A client MUST save the information of a trust message until it has fetched the corresponding key so that the key can then be authenticated or distrusted. Afterwards the information of the trust message can be deleted.
Example: Alice's endpoint A1 receives an authentication message from Bob's endpoint B1. That authentication message contains the key for Bob's other endpoint B2. If A1 has not already fetched B2's key, A1 stores the information provided by the trust message. A1 fetches B2's key and is then able to automatically authenticate A2's key.

5.3 GUI Considerations

A client that receives a trust message SHOULD NOT display its bare content to the user. Instead, the trust message SHOULD be hidden and the automatic authentication or distrusting

SHOULD take place in the background.

6 Security Considerations

6.1 Notification and Confirmation

After a successful authentication or distrusting, the user MAY be informed of that event. The client MAY offer an option for requesting the user's confirmation before any automatic authentication or automatic distrusting is performed.

6.2 Recommended Security Policy - Trust Only Authenticated Keys After First Authentication (TOAKAFA)

It is more secure to be protected against passive attacks instead of not using any encryption. If it is not possible to authenticate a key before encrypting with it but it is desired to communicate with the key's endpoint, it is RECOMMENDED to automatically trust new keys until the first authentication has been made.
Even ATM cannot protect the user against an attacker with an automatically trusted and undetected malicious key. For this reason it is important to take special care of the following security aspects.
If keys are automatically trusted until the first authentication, keys that are not authenticated by then SHOULD NOT be used any longer for encryption until they have been authenticated too. New keys SHOULD also only be used for encryption after they have been authenticated. Without these two additional precautions it is not possible to protect the user against attackers who introduced malicious keys before or after the first authentication.

7 IANA Considerations

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

8 XMPP Registrar Considerations

8.1 Protocol Namespaces

This specification defines the following XMPP namespaces:

- urn:xmpp:atm:1

8.2 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.