XEP-0364: Current Off-the-Record Messaging Usage

Sam Whited
mailto:sam@samwhited.com
xmpp:sam@samwhited.com
https://blog.samwhited.com/

2019-08-20
Version 0.3.2

Status	Type	Short Name
Deferred	Informational	NOT_YET_ASSIGNED

This document outlines the current usage of Off-the-Record messaging in XMPP, its drawbacks, its strengths, and recommendations for improving the end user experience.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2024 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction 1

2 Overview 1

3 Discovery 1

4 OTR Messages 2
- 4.1 Construction and Decoding . . . 2
- 4.2 Routing . . . 3
- 4.3 Processing Hints . . . 3
- 4.4 Explicit Message Encryption . . . 3
- 4.5 Delivery Receipts . . . 4

5 OTR Sessions 4
- 5.1 Starting an OTR session . . . 4
- 5.2 Ending an OTR session . . . 4

6 Use in XMPP URIs 5

7 Acknowledgements 5

8 Security Considerations 5

9 IANA Considerations 6

10 XMPP Registrar Considerations 6

1 Introduction

The Off-the-Record messaging protocol (OTR) was originally introduced in the 2004 paper *Off-the-Record Communication, or, Why Not To Use PGP* [1] and has since become the de facto standard for performing end-to-end encryption in XMPP. OTR provides encryption, deniable authentication, forward secrecy, and malleable encryption.
The OTR protocol itself is currently described by the document: *Off-the-Record Messaging Protocol version 3* [2] and will not be redescribed here. Instead, this document aims to describe OTR's usage and best practices within XMPP. It is not intended to be a current standard, or technical specification, as better (albeit, newer and less well tested) methods of end-to-end encryption exist for XMPP.

2 Overview

Though this document will not focus on the OTR protocol itself, a brief overview is warranted to better understand the protocols strengths and weaknesses.
OTR uses 128 bit AES symmetric-key encryption and the SHA-1 hash function. An OTR session can be held only between two parties, meaning that OTR is incompatible with Multi-User Chat (XEP-0045) [3] and Mediated Information eXchange (MIX) (XEP-0369) [4]. It provides deniability in the form of malleable encryption (a third party may generate fake messages after the session has ended). This means that if you were not a part of the original conversation, you cannot prove, based on captured messages alone, that a message from the conversation was actually sent by a given party. Unlike PGP, OTR also provides forward secrecy; even if a session is recorded and the primary key is compromised at a later date, the OTR messages will not be able to be decrypted as each was encrypted with an ephemeral key exchanged via Diffie-Hellman key exchange with a 1536 bit modulus.

3 Discovery

Clients that support the OTR protocol do not advertise it in any of the normal XMPP ways. Instead, OTR provides its own discovery mechanism. If a client wishes to indicate support for OTR they include a special whitespace tag in their messages. This tag can appear anywhere in the body of the message stanza, but it is most often found at the end. The OTR tag comprises the following bytes:

[1] Nikita Borisov, Ian Goldberg, Eric Brewer (2004-10-28). "Off-the-Record Communication, or, Why Not To Use PGP" <https://otr.cypherpunks.ca/otr-wpes.pdf>
[2] "Off-the-Record Messaging Protocol version 3" <https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html>
[3] XEP-0045: Multi-User Chat <https://xmpp.org/extensions/xep-0045.html>.
[4] XEP-0369: Mediated Information eXchange (MIX) <https://xmpp.org/extensions/xep-0369.html>.

Listing 1: OTR tag

```
\x20\x09\x20\x20\x09\x09\x09\x09 \x20\x09\x20\x09\x20\x09\x20\x20
```

and is followed by one or more of the following sequences to indicate the version of OTR which the client supports:

Listing 2: OTR tag version 1

```
\x20\x09\x20\x09\x20\x20\x09\x20
```

Note that this version 1 tag must come before other version tags for compatibility; it is, however, NOT RECOMMENDED to implement version 1 of the OTR protocol.

Listing 3: OTR tag version 2

```
\x20\x20\x09\x09\x20\x20\x09\x20
```

Listing 4: OTR tag version 3

```
\x20\x20\x09\x09\x20\x20\x09\x09
```

When a client sees this special string in the body of a message stanza it may choose to start an OTR session immediately, or merely indicate support to the user and allow the user to manually start a session. This is done by sending a message stanza containing an OTR query message in the body which indicates the supported versions of OTR. In XMPP these are most commonly version 2 and version 3, which would be indicated by a message stanza which has a body that starts with the string:

Listing 5: OTR query

```
?OTR?v23?
```

Any message which begins with the afforementioned string (note that the version number[s] may be different), postfixed with a payload should be decrypted as an OTR message. The initialization message should not contain a payload, and should just be the initialization string by itself.

4 OTR Messages

4.1 Construction and Decoding

Some clients in the wild have been known to insert XML in the <body> node of a message. Clients that support OTR should tolerate encrypted payloads which expand to unescaped XML, and treat it as plain text.

4.2 Routing

XMPP is designed so that the client needs to know very little about where and how a message will be routed. Generally, clients are encouraged to send messages to the bare JID and allow the server to route the messages as it sees fit. However, OTR requires that messages be sent to a particular resource. Therefore clients should send OTR messages to a full JID, possibly allowing the user to determine which resource they wish to start an encrypted session with. Furthermore, if a client receives a request to start an OTR session in a carboned message (due to a server which does not support the aforementioned "private" directive, or a client which does not set it), it should be silently ignored.

4.3 Processing Hints

Message Processing Hints (XEP-0334) [5] defines a set of hints for how messages should be handled by XMPP servers. These hints are not hard and fast rules, but suggestions which the servers may or may not choose to follow. Best practice is to include the following hints on all OTR messages:

```
<no-copy xmlns="urn:xmpp:hints"/>
<no-permanent-store xmlns="urn:xmpp:hints"/>
```

Similarly the "private" directive from Message Carbons (XEP-0280) [6] should also be included to indicate that carbons are not necessary (since no other resource will be able to read the message):

```
<private xmlns="urn:xmpp:carbons:2"/>
```

4.4 Explicit Message Encryption

Explicit Message Encryption (XEP-0380) [7] defines a hint to let clients without OTR support know that this message was encrypted, and display a friendly message instead of the raw encrypted data. It is RECOMMENDED that the client adds this hint alongside every encrypted message

```
<encryption xmlns="urn:xmpp:eme:0" namespace="urn:xmpp:otr:0"/>
```

All together, an example OTR message might look like this (with the majority of the body stripped out for readability):

[5] XEP-0334: Message Processing Hints <https://xmpp.org/extensions/xep-0334.html>.
[6] XEP-0280: Message Carbons <https://xmpp.org/extensions/xep-0280.html>.
[7] XEP-0380: Explicit Message Encryption <https://xmpp.org/extensions/xep-0380.html>.

Listing 6: OTR message with processing hints

```
<message from='malvolio@stewardsguild.lit/countesshousehold'
         to='olivia@countess.lit/veiled'>
  <body>?OTR?v23?...</body>
  <encryption xmlns="urn:xmpp:eme:0" namespace="urn:xmpp:otr:0"/>
  <no-copy xmlns="urn:xmpp:hints"/>
  <no-permanent-store xmlns="urn:xmpp:hints"/>
  <private xmlns="urn:xmpp:carbons:2"/>
</message>
```

4.5 Delivery Receipts

If a client supports OTR and Message Delivery Receipts (XEP-0184) [8] it is RECOMMENDED that the client send a delivery receipt only after successfully decrypting an encrypted message.

5 OTR Sessions

5.1 Starting an OTR session

Most clients today provide options to automatically start an OTR session, to manually construct a session at the users request, or to always require the use of an OTR session even if the remote client does not support OTR.
In the interest of user experience, it is NOT RECOMMENDED to start an OTR session with a previously unseen resource or one for which we do not have OTR keys cached without first discovering if the remote end supports OTR using one of the mechanisms described in the "Discovery" section of this document except in security critical contexts where user experience is not a concern.
Instead, it is RECOMMENDED to always allow the user to manually start an OTR session and to indicate that OTR is known to be available when OTR support is discovered by any of the aforementioned mechanisms.

5.2 Ending an OTR session

It is RECOMMENDED that the lifetime of OTR sessions be limited to the lifetime of the XMPP session in which the OTR session was established. If a resource associated with either end of the OTR session goes offline (a closing stream tag is received, or a fatal stream error occurs), it is RECOMMENDED that the other end terminate the OTR session.
When an XMPP session that is hosting an OTR session ends, it is RECOMMENDED that XMPP session be completely torn down before the associated OTR session is ended. For instance, when receiving a closing stream tag, clients should send their own closing stream tag (as

[8] XEP-0184: Message Delivery Receipts <https://xmpp.org/extensions/xep-0184.html>.

specified in RFC 6120 [9]), close the underlying TCP connection (or connections), and then terminate the OTR session in that order. This prevents a race condition in some clients that attempt to automatically establish an OTR session where the OTR session is torn down and then re-established by an incomming message before the XMPP session can be closed.

6 Use in XMPP URIs

RFC 5122 [10] defines a Uniform Resource Identifier (URI) and Internationalized Resource Identifier (IRI) scheme for XMPP entities, and XMPP URI Query Components (XEP-0147) [11] defines various query components for use with XMPP URI's. When an entity has an associated OTR fingerprint its URI is often formed with "otr-fingerprint" in the query string. Eg.

Listing 7: OTR Fingerprint

```
xmpp:feste@allfools.lit?otr-fingerprint=
    AEA4D503298797D4A4FC823BC1D24524B4C54338
```

The XMPP Registrar [12] maintains a registry of queries and key-value pairs for use in XMPP URIs at <https://xmpp.org/registrar/querytypes.html>. As of the date this document was authored, the 'otr-fingerprint' query string has not been formally defined and has therefore is not officially recognized by the registrar.

7 Acknowledgements

Thanks to Daniel Gultsch for his excellent article [13] on the pitfalls of implementing OTR, and to Georg Lukas and Chris Ballinger for their feedback and corrections.

8 Security Considerations

While this document describes an existing protocol which is streamed over XMPP and therefore does not introduce any new security concerns itself, it is worth mentioning a few security issues with the underlying OTR protocol:

[9] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.
[10] RFC 5122: Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP) <http://tools.ietf.org/html/rfc5122>.
[11] XEP-0147: XMPP URI Query Components <https://xmpp.org/extensions/xep-0147.html>.
[12] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.
[13] Daniel Gultsch (Retreived on 2015-07-29). "Observations on Implementing XMPP" <https://github.com/siacs/Conversations/blob/master/docs/observations.md>

Because Diffie-Hellman (D-H) key exchange is unauthenticated, the initial D-H exchange which sets up the encrypted channel is vulnerable to a man-in-the-middle attack. No sensitive information should be sent over the encrypted channel until mutual authentication has been performed inside the encrypted channel.
OTR makes use of the SHA-1 hash algorithm. While no practical attacks have been observed in SHA-1 at the time of this writing, theoretical attacks have been constructed, and attacks have been performed on hash functions that are similar to SHA-1. One cryptographer estimated that the cost of generating SHA-1 collisions was $2.77 million dollars in 2012, and would drop to $700,000 by 2015. [14]. This puts generating SHA-1 collisions well within the reach of governments, malicious organizations, and even well-funded individuals.

9 IANA Considerations

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

10 XMPP Registrar Considerations

No namespaces or parameters need to be registered with the XMPP Registrar as a result of this document.

[14] Bruce Schneier (2012-10-05). "When Will We See Collisions for SHA-1?" <https://www.schneier.com/blog/archives/2012/10/when_will_we_se.html>