XEP-0391: Jingle Encrypted Transports

Paul Schaub
mailto:vanitasvitae@riseup.net
xmpp:vanitasvitae@jabberhead.tk

2018-07-31
Version 0.1.2

Status	Type	Short Name
Deferred	Standards Track	jet

This specification defines a method that allows to use established encryption schemes for end-to-end encryption of Jingle transports.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2020 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

Jingle Encrypted Transports (JET) strives to provide a modular and easily extensible way to wrap Jingle Transports in an additional end-to-end encryption layer. The focus of this specification lays on being modular. It should be possible to extend existing Jingle use scenarios with end-to-end encryption by simply adding a JET element to the negotiation.

2 Terminology

JET uses multiple encryption layers, so it is necessary to declare a distinct denomination for the different keys involved.

Designation	Abbrevation	Usage
Transport Key	TK	(Symmetric) key that is used to encrypt/decrypt the bytestreams sent/received through Jingle transports. This key encrypts the data two entities want to exchange. Examples for TK can be found under ”Ciphers”.
Initialization Vector	IV	Initialization vector that is used together with TK.
Transport Secret	TS	Tuple formed of TK and IV.
Envelope Element	EE	Output element of an established end-to-end encryption method when encrypting TS.

3 Principle

Lets assume Romeo wants to initiate an encrypted Jingle session with Juliet. Prior to the Jingle session initiation, an already existing, established and (ideally) authenticated end-to-end encryption session between Romeo and Juliet MUST exist. This session is needed to transfer the Transport Secret from Romeo to Juliet.
When this precondition is met, Romeo initially generates a transport key (TK) and associated initialization vector (IV). These will later be used by the sender to encrypt, and respectively by the recipient to decrypt data that is exchanged. This protocol defines a set of usable ciphers from which Romeo might choose. TK and IV together form the transport secret (TS).
Next Romeo uses his established encryption session with Juliet to encrypt TS. The resulting

envelope element (EE) will be part of the Jingle session initiation as child of the JET <security/> element.
When Juliet receives Romeos session request, she decrypts EE to retrieve TS, from which she can obtain TK and IV. Now she and Romeo can go on with the session negotiation. Once the session is established, data can be encrypted and exchanged. Both parties MUST keep a copy of TS in cache until the Jingle session is ended.

4 Encrypted Jingle File Transfer using JET

Jingle File Transfer (XEP-0234) [1] has the disadvantage, that transmitted files are not encrypted (aside from regular TLS transport encryption), which means that intermediate nodes like XMPP/proxy server(s) have access to the transferred data. Considering that end-to-end encryption becomes more and more important to protect free speech and personal expression, this is a major flaw that needs to be addressed.
In order to initiate an encrypted file transfer, the initiator includes a JET <security/> element in the Jingle file transfer request.

4.1 File Offer

In this scenario Romeo wants to send an encrypted text file over to Juliet. First, he generates a fresh AES-256 transport key and IV. In this case TK and IV are serialized into TS which is then encrypted using Romeos end-to-end-encryption session with Juliet.
The resulting envelope element (EE) is sent as part of the security element along with the rest of the jingle stanza over to Juliet.

Listing 1: Romeo initiates an encrypted file offer

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='nzu25s8'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.example/dr4hcr0st3lup4c'
          sid='851ba2'>
    <content creator='initiator' name='a-file-offer' senders='
        initiator'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <date>1969-07-21T02:56:15Z</date>
          <desc>This is a test. If this were a real file...</desc>
          <media-type>text/plain</media-type>
```

[1] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

```
          <name>test.txt</name>
          <range/>
          <size>6144</size>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
      </transport>
      <security xmlns='urn:xmpp:jingle:jet:0'
                name='a-file-offer'
                cipher='urn:xmpp:ciphers:aes-256-gcm-nopadding'
                type='urn:xmpp:encryption:stub:0'>
        <encrypted xmlns='urn:xmpp:encryption:stub:0'>
          <payload>BASE64-ENCODED-ENCRYPTED-SECRET</payload>
        </encrypted>
      </security>
    </content>
  </jingle>
</iq>
```

Juliet decrypts the envelope element (EE) using her session with Romeo to retrieve TS from which she deserializes TK and IV. Both Juliet and Romeo then carry on with the session negotiation as described in Jingle File Transfer (XEP-0234) [2]. Before Romeo starts transmitting the file, he encrypts it using TK and IV. He then transmitts the encrypted file over to Juliet. When Juliet received the file, she uses the TK and IV to decrypt the received file.

4.2 File Request

Juliet might want to request a file transfer from Romeo. This can be the case, when Romeo hosts the file. In order to do so, she sends generates TK and IV, creates TS from those and encrypts TS with an encryption method of her choice to get EE. TK and IV will be used by Romeo to encrypt the requested file before sending it to Juliet.

Listing 2: Juliet initiates an encrypted file request

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
```

[2] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

```
    id='wsn361c3'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='juliet@capulet.example/yn0cl4bnw0yr3vym'
          sid='uj3b2'>
    <content creator='initiator' name='a-file-request' senders='
        responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
                 mode='tcp'
                 sid='xig361fj'>
        <candidate cid='ht567dq'
                   host='192.169.1.10'
                   jid='juliet@capulet.example/yn0cl4bnw0yr3vym'
                   port='6539'
                   priority='8257636'
                   type='direct'/>
      </transport>
      <security xmlns='urn:xmpp:jingle:jet:0'
                name='a-file-request'
                cipher='urn:xmpp:ciphers:aes-256-gcm-nopadding'
                type='urn:xmpp:encryption:stub:0'>
        <encrypted xmlns='urn:xmpp:encryption:stub:0'>
          <payload>BASE64-ENCODED-ENCRYPTED-SECRET</payload>
        </encrypted>
      </security>
    </content>
  </jingle>
</iq>
```

4.3 Encrypted Ranged Transfers

Jingle File Transfer (XEP-0234) [3] defines a way for parties to request ranged transfers. This can be used to resume interrupted transfers etc. In case of an interrupted transfer, the receiving party might be able to decrypt parts of the received file. When requesting a resumption of the transfer, the recipient therefore can use the index of the last successfully decrypted byte of the file as offset in the ranged transfer. Since a resumed transfer takes place in a new session, the old transport secret might no longer be available to either party. For that reason the receiver creates a new TS for the session-initiation. The sending party then encrypts and

[3] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

sends only the requested parts of the file.

Listing 3: Romeo requests the resumption of an interrupted transfer using a fresh transport secret

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='wsn361c3'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-initiate'
          initiator='romeo@montague.example/dr4hcr0st3lup4c'
          sid='uj3b2'>
    <content creator='initiator' name='restart' senders='responder'>
      <description xmlns='urn:xmpp:jingle:apps:file-transfer:5'>
        <file>
          <range offset='270336'/>
          <hash xmlns='urn:xmpp:hashes:2'
                algo='sha-1'>w0mcJylzCn+AfvuGdqkty2+KP48=</hash>
        </file>
      </description>
      <transport xmlns='urn:xmpp:jingle:transports:s5b:1'
            mode='tcp'
                 sid='vj3hs98y'>
        <candidate cid='hft54dqy'
                   host='192.168.4.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5086'
                   priority='8257636'
                   type='direct'/>
        <candidate cid='hutr46fe'
                   host='24.24.24.1'
                   jid='romeo@montague.example/dr4hcr0st3lup4c'
                   port='5087'
                   priority='8258636'
                   type='direct'/>
      </transport>
      <security xmlns='urn:xmpp:jingle:jet:0'
                name='restart'
                cipher='urn:xmpp:ciphers:aes-256-gcm-nopadding'
                type='urn:xmpp:encryption:stub:0'>
        <encrypted xmlns='urn:xmpp:encryption:stub:0'>
          <payload>BASE64-ENCODED-ENCRYPTED-SECRET</payload>
        </encrypted>
      </security>
    </content>
  </jingle>
</iq>
```

5 Ciphers

In order to encrypt the transported bytestream, the initiator must transmit a cipher key to the responder. There are multiple options available:

Namespace	Type	Length (bits)	Parameters
urn:xmpp:ciphers:aes-128-gcm-nopadding:0	AES	Key: 128, IV: 96	GCM/NoPadding
urn:xmpp:ciphers:aes-256-gcm-nopadding:0	AES	Key: 256, IV: 96	GCM/NoPadding

6 Determining Support

To advertise its support for the Jingle Encrypted Transports, when replying to service discovery information ("disco#info") requests an entity MUST return URNs for any version, or extension of this protocol that the entity supports -- e.g., "urn:xmpp:jingle:jet:0" for this version, or "urn:xmpp:jingle:jet-stub:0" for a stub encryption method (see Namespace Versioning regarding the possibility of incrementing the version number).

Listing 4: Service discovery information request

```
<iq from='romeo@montague.example/dr4hcr0st3lup4c'
    id='uw72g176'
    to='juliet@capulet.example/yn0cl4bnw0yr3vym'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

Listing 5: Service discovery information response

```
<iq from='juliet@capulet.example/yn0cl4bnw0yr3vym'
    id='uw72g176'
    to='romeo@montague.example/dr4hcr0st3lup4c'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:jingle:jet:0'/>
    <feature var='urn:xmpp:jingle:jet-stub:0'/>
  </query>
</iq>
```

In order for an application to determine whether an entity supports this protocol, where possible it SHOULD use the dynamic, presence-based profile of service discovery defined in Entity Capabilities (XEP-0115) [4]. However, if an application has not received entity capabilities

[4] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.

information from an entity, it SHOULD use explicit service discovery instead.

7 Security Considerations

The initiator SHOULD NOT use the generated key TK as IV, but instead generate a seperate random IV.
Instead of falling back to unencrypted transfer in case something goes wrong, implementations MUST instead abort the Jingle session, informing the user.
IMPORTANT: This approach does not deal with metadata. In case of Jingle File Transfer (XEP-0234) [5], an attacker with access to the sent stanzas can for example still see the name of the file and other information included in the <file/> element.
The responder MUST check, whether the envelope element belongs to the initiator to prevent MitM attacks

8 Acknowledgements

Big thanks to Florian Schmaus for mentoring my Google Summer of Code project, which resulted in this protocol. Also thanks to Andrey Gursky, Daniel Gultsch, Dave Cridland, Goffi, Jonas Wielicki and Sam Whited for their input and feedback.

[5] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.