XEP-0187: Offline Encrypted Sessions

Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

2007-05-30
Version 0.5

Status	Type	Short Name
Deferred	Standards Track	TO BE ASSIGNED

This document specifies an end-to-end encryption protocol for offline XMPP communication sessions.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction **1**

2 Dramatis Personae **2**

3 Offline ESession Negotiation **2**
- 3.1 Publishing Offline ESession Options 2
- 3.2 Requesting Offline ESession Options 6
- 3.3 Starting an Offline ESession 7
- 3.4 Accepting Offline ESessions 9

4 Exchanging Stanzas **10**

5 ESession Termination **11**

6 Security Considerations **11**
- 6.1 Replay Attacks 11
- 6.2 Options Expiry Time 11
- 6.3 Identity Protection 12
- 6.4 Conclusion 12

7 IANA Considerations **12**

8 XMPP Registrar Considerations **12**
- 8.1 Protocol Namespaces 12

1 Introduction

The convenience of sending stanzas to other entities that are offline ("offline messages") is an important and popular feature of most XMPP implementations (see Best Practices for Handling Offline Messages (XEP-0160) [1]). Without offline messages delivery would have to wait until both entities managed to be online at the same time. So many urgent messages could not be delivered in time. For example, the sender might want to send an urgent message before jumping on a flight.
End-to-end encryption is another desirable feature for any communication technology. Unfortunately it is not possible to make offline encryption quite so secure as online encryption (see Security Considerations). However, offline encryption has a long history and is certainly preferable to having no encryption at all. [2] Unfortunately, for reasons described in Cryptographic Design of Encrypted Sessions (XEP-0188) [3], the existing proposals (including Current Jabber OpenPGP Usage (XEP-0027) [4] and RFC 3923 [5]) have not been widely implemented and deployed. This document describes a different approach to offline end-to-end encryption for use by entities that communicate using XMPP. It builds on the existing online Encrypted Session Negotiation (XEP-0116) [6] protocol. As a result it offers important advantages over the existing "Object Encryption" proposals:

- Perfect Forward Secrecy [7]
- Other security features described in Cryptographic Design of Encrypted Sessions
- Most of the code can be borrowed from implementations of Encrypted Session Negotiation

The requirements and the consequent cryptographic design that underpin this protocol are described in Requirements for Encrypted Sessions (XEP-0210) [8] and Cryptographic Design of Encrypted Sessions. The basic concept is that of an encrypted session which acts as a secure tunnel between the online endpoint and the offline endpoint. The offline endpoint completes its part of the tunnel "negotiation" by publishing its preferences before it goes offline. Once the tunnel is established by the online endpoint, the content of each one-to-one XML stanza sent by the online endpoint is encrypted and then transmitted within a "wrapper" stanza using Stanza Encryption (XEP-0200) [9].

[1] XEP-0160: Best Practices for Handling Offline Messages <https://xmpp.org/extensions/xep-0160.html>.
[2] This protocol does not stop paranoid users avoiding sending offline messages.
[3] XEP-0188: Cryptographic Design of Encrypted Sessions <https://xmpp.org/extensions/xep-0188.html>.
[4] XEP-0027: Current Jabber OpenPGP Usage <https://xmpp.org/extensions/xep-0027.html>.
[5] RFC 3923: End-to-End Signing and Object Encryption for the Extensible Messaging and Presence Protocol (XMPP) <http://tools.ietf.org/html/rfc3923>.
[6] XEP-0116: Encrypted Session Negotiation <https://xmpp.org/extensions/xep-0116.html>.
[7] Long-lived keys are typically used for a few years, whereas Offline ESession decryption keys typically exist for just a few hours. So the Perfect Forward Secrecy feature significantly enhances the security of Offline ESessions.
[8] XEP-0210: Requirements for Encrypted Sessions <https://xmpp.org/extensions/xep-0210.html>.
[9] XEP-0200: Stanza Encryption <https://xmpp.org/extensions/xep-0200.html>.

2 Dramatis Personae

This document introduces two characters to help the reader follow the necessary exchanges:

1. "Bob" is the name of the online participant in the offline ESession. Within the scope of this document, his fully-qualified JID is: <bob@example.com/laptop>.
2. "Alice" is the name of the offline participant in the offline ESession started by Bob. Within the scope of this document, we stipulate that her fully-qualified JID is: <alice@example.org/pda>.

While Bob and Alice are introduced as "end users", they are simply meant to be examples of Jabber entities. Any directly addressable Jabber entity may participate in an offline ESession.

3 Offline ESession Negotiation

The approach to establishing a secure session with an entity that is offline is in essence a special case of 3-message negotiation employed for online ESessions (see Encrypted Session Negotiation).
Alice MAY publish a set of offline ESession options just before she goes offline. If Bob subscribes to Alice's presence and wishes to establish an online ESession with her, but he finds that she is offline, then if she published her offline ESession options before going offline, he may use the protocol described below to perform the initial Diffie-Hellman key exchange, establish an offline ESession and send encrypted stanzas to her while she is offline. Note: Bob MUST NOT initiate a new ESession with Alice if he already has an ESession established with her.
Note: Alice MAY also publish *another* similar set of relatively long-lived [10] offline ESession options that any entity MAY use for the same purpose.

3.1 Publishing Offline ESession Options

In order to publish either set of her offline ESession options Alice MUST create an options data form in exactly the same way as she would create an online ESession request data form (see the ESession Request section in Encrypted Session Negotiation) except she MUST omit The 'accept' and 'pubkey' fields. Note: The list of stanza types she is willing to decrypt MUST NOT include the value 'iq'.
Alice MUST append to the content of the form an 'expires' field containing the UTC time (see XMPP Date and Time Profiles (XEP-0082) [11]) that she decides her offline ESession options will

[10] The more often Alice changes her published ESession options, the shorter the Perfect Forward Secrecy window of vulnerability. However, whenever she changes them she divulges her presence to all the entities that are monitoring them.
[11] XEP-0082: XMPP Date and Time Profiles <https://xmpp.org/extensions/xep-0082.html>.

expire (see Options Expiry Time Security Considerations).
Alice MUST store her value of NA (her ESession ID), all her values of x (one for each MODP group) and the time she decides her offline ESession options will expire in a secure way, so that she can retrieve them when she comes back online (idealy even if that is using a different client and/or a different machine).
If Alice would not be able to decrypt stanzas if she came back online using a different client and/or a different machine then she SHOULD also encapsulate the resource of her client in a 'match_resource' field and append it to her options data form. In this case, the list of stanza types she is willing to decrypt MUST include only 'message'.
Alice MUST also append to the content of the form a list of signatures (see Signature Generation) of the *content* of the data form (excluding the 'signs' field). The list SHOULD include one signature for each of the public signature-verification keys that other entities may use to authenticate her.
Alice MUST publish the ESession options data form through her own server using Personal Eventing Protocol (XEP-0163) [12].
If the pubkey PEP node does not exist already then Alice MUST create it first. In this case, Alice SHOULD specify the "Presence" access model for the set of options for presence subscribers (or the "Open" model for the set for other entities), unless she wants greater control over access to her identity (see Identity Protection Security Considerations). Alice SHOULD specify that the options will never be pushed to subscribers (even when she publishes new options) - especially if she specifies the "Whitelist" access model.

Listing 1: Alice Creates PEP Node for ESession Options for Presence Subscribers

```
<iq from='alice@example.org/pda' type='set' id='create1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='http://www.xmpp.org/extensions/xep-0187.html#ns'/>
    <configure>
      <x xmlns='jabber:x:data' type='form'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#node_config</value>
        </field>
        <field var='pubsub#deliver_notifications'>
          <option><value>0</value></option>
        </field>
        <field var='pubsub#send_last_published_item'>
          <option><value>never</value></option>
        </field>
        <field var='pubsub#access_model'>
          <option><value>presence</value></option>
        </field>
      </x>
    </configure>
  </pubsub>
</iq>
```

[12] XEP-0163: Personal Eventing Protocol <https://xmpp.org/extensions/xep-0163.html>.

Listing 2: Alice Creates PEP Node for ESession Options for All Entities

```
<iq from='alice@example.org/pda' type='set' id='create2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <create node='http://www.xmpp.org/extensions/xep-0116.html#ns'/>
    <configure>
      <x xmlns='jabber:x:data' type='form'>
        <field var='FORM_TYPE' type='hidden'>
          <value>http://jabber.org/protocol/pubsub#node_config</value>
        </field>
        <field var='pubsub#deliver_notifications'>
          <option><value>0</value></option>
        </field>
        <field var='pubsub#send_last_published_item'>
          <option><value>never</value></option>
        </field>
        <field var='pubsub#access_model'>
          <option><value>open</value></option>
        </field>
      </x>
    </configure>
  </pubsub>
</iq>
```

Once the publishing node has been created, Alice can update her options for subscribers to her presence whenever she goes offline:

Listing 3: Alice Publishes Her ESession Options for Her Presence Subscribers

```
<iq from='alice@example.org/pda' type='set' id='pub1'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='http://www.xmpp.org/extensions/xep-0187.html#ns'>
      <item>
        <x type='form' xmlns='jabber:x:data'>
          <field type="hidden" var="FORM_TYPE">
            <value>urn:xmpp:ssn</value>
          </field>
          <field type="list-single" var="otr">
            <option><value>false</value></option>
            <option><value>true</value></option>
          </field>
          <field type="list-single" var="disclosure">
            <option><value>never</value></option>
          </field>
          <field type="list-single" var="security">
            <option><value>e2e</value></option>
          </field>
          <field type="list-single" var="modp">
            <option><value>5</value></option>
            <option><value>14</value></option>
```

```
          <option><value>2</value></option>
        </field>
        <field type="list-single" var="crypt_algs">
          <option><value>aes256-ctr</value></option>
          <option><value>twofish256-ctr</value></option>
          <option><value>aes128-ctr</value></option>
        </field>
        <field type="list-single" var="hash_algs">
          <option><value>whirlpool</value></option>
          <option><value>sha256</value></option>
        </field>
        <field type="list-single" var="sign_algs">
          <option><value>rsa</value></option>
          <option><value>dsa</value></option>
        </field>
        <field type="list-single" var="compress">
          <option><value>none</value></option>
        </field>
        <field type="list-multi" var="stanzas">
          <option><value>message</value></option>
        </field>
        <field type="list-single" var="ver">
          <option><value>1.3</value></option>
          <option><value>1.2</value></option>
        </field>
        <field type="text-single" var="rekey_freq">
          <value>1</value>
        </field>
        <field var="expires">
          <value>2006-06-09T02:56:15Z</value>
        </field>
        <field var="my_nonce">
          <value> ** Base64 encoded ESession ID ** </value>
        </field>
        <field var="dhkeys">
          <value> ** Base64 encoded value of e5 ** </value>
          <value> ** Base64 encoded value of e14 ** </value>
          <value> ** Base64 encoded value of e2 ** </value>
        </field>
        <field var="match_resource">
          <value>pda</value>
        </field>
        <field var="signs">
          <value> ** signature of form ** </value>
          <value> ** signature of form ** </value>
        </field>
      </x>
    </item>
  </publish>
```

```
  </pubsub>
</iq>
```

At the risk of divulging her presence, Alice MAY also update her options for all entities:

Listing 4: Alice Publishes Her ESession Options for All Entities

```
<iq type='set' from='alice@example.org/pda' id='pub2'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='http://www.xmpp.org/extensions/xep-0116.html#ns'>
      ...
    </publish>
  </pubsub>
</iq>
```

3.2 Requesting Offline ESession Options

If Bob believes Alice is offline he SHOULD request her ESession options and, if he does not have a local copy of any of her public keys, her long-term public signature-verification keys from her server using Personal Eventing via Pubsub (see Public Key Publishing (XEP-0189) [13]). [14]

If Bob is subscribing to Alice's presence he MUST request her ESession Options exclusively for subscribers.

Listing 5: Bob asks Alice's Server for her ESession Options (Subscribers)

```
<iq type='get'
    from='bob@example.com/laptop'
    to='alice@example.org'
    id='es4'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='http://www.xmpp.org/extensions/xep-0187.html#ns'/>
  </pubsub>
</iq>
```

If Bob is not subscribing to Alice's presence (or if Alice has no ESession options exclusively for subscribers) he MUST use the following request instead.

Listing 6: Bob asks Alice's Server for her ESession Options (All Entities)

```
<iq type='get'
    from='bob@example.com/laptop'
    to='alice@example.org'
```

[13] XEP-0189: Public Key Publishing <https://xmpp.org/extensions/xep-0189.html>.

[14] There is no need for Bob to discover Alice's support for the Offline ESessions protocol via Service Discovery (XEP-0030) [15].

```
    id='es4'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <items node='http://www.xmpp.org/extensions/xep-0116.html#ns'/>
  </pubsub>
</iq>
```

If, after receiving Alice's offline ESession options, Bob is unable to verify any of Alice's signatures of her options data form (see Signature Verification) with any of her public keys already in his possession (or with any of her keys he is able to acquire and trust) then he MUST proceed no further, since he cannot be sure who will be able to decrypt his stanzas.
If Bob cannot acquire Alice's ESession options, or he does not support one or more of the options in each Alice's ESession fields, or if the 'expires' field indicates that Alice's options have expired, then he MUST NOT send encrypted stanzas to her while she is offline.

3.3 Starting an Offline ESession

Bob MUST now continue as if Alice had requested an online ESession, performing the steps described in three of the sections of Encrypted Session Negotiation:

1. Diffie-Hellman Preparation (Bob) Note: If the value of e he selected is not valid, Bob SHOULD terminate the ESession *without* sending an error.

2. Generating Session Keys

3. Hiding Identity Note: Since Bob did not receive a 'pubkey' field, he MUST assume its value is 'key'. Bob SHOULD NOT include a 'pubkey' field in formB since Alice has already 'proved' her identity.

As with 3-message ESession negotiation, Bob should encapsulate the Base64 encoded values of IDB and MB in data form fields ('identity' and 'mac'), and append the new fields to formB.
Bob MAY also send encrypted content (see the Exchanging Stanzas section of Encrypted Session Negotiation) in the same stanza. Note: If he also includes a field named "terminate" set to a value of "1" or "true" within the data form (see the ESession Termination section of Encrypted Session Negotiation) then the ESession is terminated immediately. This special case, where a single stanza is encrypted and sent in isolation, is equivalent to object encryption (or object signing if no encryption is specified) and offers several significant advantages over non-session approaches - including perfect forward secrecy.
If Alice included a 'match_resource' field in her ESession options, then Bob MUST address all the stanzas he sends within the offline ESession to the specified resource and use the Advanced Message Processing protocol to ensure that they are not delivered to any other resource.

After sending formB to Alice, Bob can assume that the ESession negotiation is complete.

Listing 7: Bob Establishes an ESession Without Negotiation

```
<message from='bob@example.com/laptop' to='alice@example.org/pda' type
    ='chat'>
  <init xmlns='http://www.xmpp.org/extensions/xep-0116.html#ns-init'>
    <x type='submit' xmlns='jabber:x:data'>
      <field var="FORM_TYPE">
        <value>urn:xmpp:ssn</value>
      </field>
      <field var="otr"><value>true</value></field>
      <field var="disclosure"><value>never</value></field>
      <field var="security"><value>e2e</value></field>
      <field var="modp"><value>5</value></field>
      <field var="crypt_algs"><value>aes256-ctr</value></field>
      <field var="hash_algs"><value>sha256</value></field>
      <field var="sign_algs"><value>rsa</value></field>
      <field var="compress"><value>none</value></field>
      <field var="stanzas"><value>message</value></field>
      <field var="ver"><value>1.3</value></field>
      <field var="rekey_freq"><value>50</value></field>
      <field var="my_nonce">
        <value> ** Base64 encoded ESession ID ** </value>
      </field>
      <field var="dhkeys">
        <value> ** Base64 encoded value of d ** </value>
      </field>
      <field var="nonce">
        <value> ** Base64 encoded ESession ID ** </value>
      </field>
      <field var="counter">
        <value> ** Base64 encoded block counter ** </value>
      </field>
      <field var="identity">
        <value> ** Encrypted identity ** </value>
      </field>
      <field var="mac">
        <value> ** Integrity of identity ** </value>
      </field>
    </x>
  </init>
  <c xmlns='http://www.xmpp.org/extensions/xep-0200.html#ns'>
    <data> ** Base64 encoded m_final ** </data>
    <mac> ** Base64 encoded a_mac ** </mac>
  </c>
  <amp xmlns='http://jabber.org/protocol/amp'>
    <rule action='??????' condition='match-resource' value='exact'/>
  </amp>
```

```
</message>
```

3.4 Accepting Offline ESessions

When Alice comes online she MUST perform the following steps:

1. Ensure she is no longer publishing offline ESession options exclusively for entities that are subscribing to her presence.

Listing 8: Alice Stops Publishing Her ESession Options

```
<iq type='set' from='alice@example.org/pda' id='es5'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='http://www.xmpp.org/extensions/xep-0187.html#
        ns'>
      <item/>
    </publish>
  </pubsub>
</iq>
```

2. Retrieve any values of NA, x (one for each MODP group for each NA) and her offline ESession options expiry time that she stored before going offline, and destroy in a secure way any persistently stored copies that correspond to ESession options exclusively for subscribers.

If the current time is after her offline ESession options expiry time then she MUST discard all stanzas from offline ESessions without decrypting them. Otherwise, when Alice receives an offline ESession request stanza from Bob then she MUST perform the following steps:

1. Select her value of x that corresponds to the 'nonce' and 'modp' fields she received from Bob. [16]

2. Confirm that she has not already received a key exchange stanza from Bob with the same value of d or NB ('my_nonce' field) since she published her ESession options (see the Replay Attacks subsection of the Security Considerations section of Encrypted Session Negotiation). If the options were for subscribers, that means since she came online.

[16] Alice may not be able to find the specified value of x if, for example, she went offline using a different client and/or a different machine without publishing a 'match_resource' field. In this case Alice cannot decrypt the offline ESession!

3. Alice MUST now continue as if Bob had responded to her online ESession request, performing the steps described in two of the sections of Encrypted Session Negotiation:

   - Diffie-Hellman Preparation (Alice) Note: If she is not prepared to support any of the ESession options specified by Bob, or if the value of d she selected is not valid, then Alice SHOULD terminate the ESession *without* sending an error.

   - Verifying Bob's Identity Note: Since Alice did not send a 'pubkey' field to Bob, she MUST assume its value is 'key'. If the value of MB she calculated does not match the one she received, or if she cannot confirm that pubKeyB really is Bob's public key, or if she cannot confirm that signB is the signature of the HMAC result, then Alice SHOULD terminate the ESession *without* sending an error.

4 Exchanging Stanzas

Alice MUST NOT send encrypted content within an offline ESession started by Bob. If Bob is conducting an offline ESession with Alice when she is online (e.g., if he is not subscribing to her presence), then if Alice wants to send a stanza to Bob, she MUST terminate the offline ESession and start a new online ESession first.
For Offline ESessions, Bob SHOULD include a 'Created' SHIM header in the encrypted content. Assuming she trusts Bob, Alice SHOULD trust this header and ignore the unencrypted Legacy Delayed Delivery (XEP-0091) [17] element inserted by her server.

Listing 9: Unencrypted Stanza

```
<message from='bob@example.com/laptop'
         to='alice@example.org/pda'
         type='chat'>
  <body>Hello, Alice!</body>
  <amp xmlns='http://jabber.org/protocol/amp'>
    <rule action='error' condition='match-resource' value='exact'/>
  </amp>
  <headers xmlns='http://jabber.org/protocol/shim'>
    <header name='Created'>2005-02-10T03:01:52Z</header>
  </headers>
  <active xmlns='http://jabber.org/protocol/chatstates'/>
</message>
```

Listing 10: XML Content to be Encrypted

```
<body>Hello, Alice!</body>
<headers xmlns='http://jabber.org/protocol/shim'>
```

[17] XEP-0091: Legacy Delayed Delivery <https://xmpp.org/extensions/xep-0091.html>.

```
  <header name='Created'>2005-02-10T03:01:52Z</header>
</headers>
<active xmlns='http://jabber.org/protocol/chatstates'/>
```

5 ESession Termination

If Bob notices that Alice comes online during his offline ESession with her then he MUST terminate the ESession immediately. If required he may then negotiate a new (more secure) online ESession.
When Alice receives the offline ESession termination stanza from Bob, she SHOULD NOT send a termination stanza back to him.

6 Security Considerations

For more security considerations refer to the Security Considerations section of Encrypted Session Negotiation

6.1 Replay Attacks

Since Alice supplies the same set of values of e for all offline ESessions, to prevent complete offline ESessions being replayed to her, she MUST take care to securely store *new* values (or destroy existing values) of NA and x for subscribers whenever she goes offline (see Publishing ESession Options). Also, when Alice comes online again, she MUST remember all the values of d he receives in offline ESession negotiation stanzas, and reject any offline ESessions that specify a value of d she has already received (see Accepting an Offline ESession).
Note: If Alice publishes ESession options for non-subscribers, and if she does not update them whenever she comes online then, until she updates the options, she MUST save all the values of d she receives to secure persistent storage (along with the values of NA and x).

6.2 Options Expiry Time

If Alice's offline private Diffie-Hellman keys are compromised before she destroys them, and the attacker also controls communications between Bob and Alice's server, then the attacker could prevent Bob discovering if she comes online, and resend her compromised ESession options to Bob whenever he asks for them. This would allow the attacker to decrypt all messages sent to Alice before her offline ESession options expire. Alice would probably never receive the messages sent to her after she comes back online. If an attack is discovered before the compromised ESession options expire then Alice SHOULD change her long-term signing key. Alice SHOULD mitigate this attack by choosing an expiry time for her ESession options

that is not too long after she expects to be online again.

6.3 Identity Protection

Alice's Offline ESession options include her signatures. Although the signatures are public information, any entity that possesses one of Alice's corresponding public keys may confirm whether or not she is the entity using the JID. If it is important for Alice to protect her identity (i.e. hide the fact that she is the user of the JID), then she MUST carefully control who has access to her public keys.

6.4 Conclusion

In addition to the issues above, the Perfect Forward Secrecy window of vulnerability is longer. More seriously, Alice MUST store her private Diffie-Hellman keys, x1...xZ, to local disk or to a server (perhaps symmetrically encrypted with a password). It is *really* hard to securely erase something from a disk. Note: Once compromised, her private Diffie-Hellman keys could only be used to decrypt messages sent to her within offline Esessions until her offline ESession options expire. They could not be used to negotiate online ESessions [18], or to send messages that appear to be from Alice. This is a significant benefit when compared to the existing "object encryption" schemes.
Alice could decide not to support Offline ESessions since they are less secure than online ESessions. However, if Alice does that then, while she is offline, Bob, or Aunt Tillie, will probably send her completely unprotected messages!

7 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [19].

8 XMPP Registrar Considerations

8.1 Protocol Namespaces

Until this specification advances to a status of Draft, its associated namespace shall be "http://www.xmpp.org/extensions/xep-0187.html#ns"; upon advancement of this specifica-

[18] The online ESessions protocol forces both Alice and Bob to produce a signature of both their Diffie-Hellman keys together.

[19] 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/>.

tion, the XMPP Registrar [20] shall issue a permanent namespace in accordance with the process defined in Section 4 of XMPP Registrar Function (XEP-0053) [21].

[20] 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/>.

[21] XEP-0053: XMPP Registrar Function <https://xmpp.org/extensions/xep-0053.html>.