XEP-0185: Dialback Key Generation and Validation

Philipp Hancke
mailto:fippo@goodadvice.pages.de
xmpp:fippo@goodadvice.pages.de

Peter Saint-Andre
mailto:xsf@stpeter.im
xmpp:peter@jabber.org
http://stpeter.im/

2007-02-15
Version 1.0

Status	Type	Short Name
Active	Informational	N/A

This document provides a recommended method for generating and validating the keys used in the XMPP server dialback protocol.

Legal

Copyright

Permissions

Warranty

Liability

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

RFC 3920 [1] does not specify in detail a recommended algorithm for generating the keys used in the server dialback protocol. This document provides such a recommendation as an aid to implementors of XMPP servers. This document is not meant to supersede any text in RFC 3920; however, the recommendations in this document have been incorporated into Server Dialback (XEP-0220) [2].

2 Recommended Algorithm

The process for generating and validating a dialback key SHOULD take into account the following four inputs:

- the hostname of the Originating Server
- the hostname of the Receiving Server
- the Stream ID generated by the Receiving Server
- a secret known by the Authoritative Server's network

In particular, the following algorithm is RECOMMENDED:

```
key = HMAC-SHA256
      (
        SHA256(Secret),
        {
          Receiving Server, '␣',
          Originating Server, '␣',
          Stream ID
        }
      )
```

Note the following:

1. The resulting dialback key is a Keyed-Hash Message Authentication Code (see HMAC [3]) generated using the SHA256 hashing algorithm (see SHA [4]).

[1] RFC 3920: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc3920>.
[2] XEP-0220: Server Dialback <https://xmpp.org/extensions/xep-0220.html>.
[3] The Keyed-Hash Message Authentication Code (HMAC): Federal Information Processing Standards Publication 198 <http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf>.
[4] Secure Hash Standard: Federal Information Processing Standards Publication 180-2 <http://csrc.nist.gov/publications/fips/fips180-2/fips186-2withchangenotice.pdf>.

2. The Secret is used as a ”key” within the HMAC generation process; because HMAC recommends that the length of the HMAC key should be at least half the size of the hash function output, the Secret SHOULD be hashed via SHA256 prior to use in the in HMAC generation process.

3. The Secret MUST either be set up in a configuration option for each host or process, or generated as a random string when starting the XMPP server. Creation of the Secret MUST NOT require communication between the Originating Server, the Authoritative Server, and optionally a third party (such as a database).

4. The output of the SHA256 hashing algorithm MUST be provided in the hexadecimal representation; this helps to avoid encoding problems.

5. The hostname of the Receiving Server, the hostname of the Originating Server, and the Stream ID SHOULD be concatenated with a Unicode space character (U+0020) as the delimiter; this helps to avoid ambiguity in concatenation. [5]

3 Example

This document closely follows the description of the dialback protocol in RFC 3920 and XEP-0220, but omits steps that are not important for the generation and validation of the dialback keys. For ease of comparison the numbering of the steps is the same as in section 8.3 of RFC 3920 and Appendix C.3 of XEP-0220. Any line breaks in the examples are included for the purpose of readability only.
The following data values are used in the examples:

Originating Server	example.org
Authoritative Server	example.org
Receiving Server	xmpp.example.com
Secret	s3cr3tf0rd14lb4ck
Stream ID	D60000229F

3. Receiving Server sends a stream header back to the Originating Server, including a unique ID for this interaction:

```
<stream:stream
  xmlns:stream='http://etherx.jabber.org/streams'
  xmlns='jabber:server'
  xmlns:db='jabber:server:dialback'
```

[5] For example, the string ”example.inform.example.org” could be construed as a concatenation of ”example.info” and ”rm.example.org” or of ”example.in” and ”form.example.org”.

```
  to='xmpp.example.com'
  from='example.org'
  id='D60000229F'>
```

The Originating Server now generates a dialback key to be sent to the Receiving Server:

```
key = HMAC-SHA256(
                  SHA256(secret),
                  { Receiving server, ' ', Originating server, ' ',
                      Stream ID}
                )
    = HMAC-SHA256(
                  SHA256('s3cr3tf0rd14lb4ck'),
                  { 'xmpp.example.com', ' ', 'example.org', ' ', '
                      D60000229F' }
                )
    = HMAC-SHA256(
                  '
                      a7136eb1f46c9ef18c5e78c36ca257067c69b3d518285f0b18a96c33beae9ac
                      ',
                  'xmpp.example.com example.org D60000229F'
                )
    = '37
       c69b1cf07a3f67c04a5ef5902fa5114f2c76fe4a2686482ba5b89323075643
       '
```

4. The Originating Server sends the generated dialback key to the Receiving Server:

```
<db:result
  to='xmpp.example.com'
  from='example.org'>
    37c69b1cf07a3f67c04a5ef5902fa5114f2c76fe4a2686482ba5b89323075643
</db:result>
```

8. The Receiving Server sends the Authoritative Server a request for verification of the key:

```
<db:verify
  to='example.org'
  from='xmpp.example.com'
  id='D60000229F'>
    37c69b1cf07a3f67c04a5ef5902fa5114f2c76fe4a2686482ba5b89323075643
</db:verify>
```

The Authoritative Server calculates the valid key for this verify request, using data supplied in the packet and the secret shared with the Originating Server.

```
key = HMAC-SHA256(
```

```
                    SHA256(secret),
                    { Receiving Server, '␣', Authoritative Server, '␣',
                        Stream ID }
                )
    = HMAC-SHA256(
                    SHA256('s3cr3tf0rd14lb4ck'),
                    { 'xmpp.example.com', '␣', 'example.org', '␣', '
                        D60000229F' }
                )
    = HMAC-SHA256(
                     '
                         a7136eb1f46c9ef18c5e78c36ca257067c69b3d518285f0b18a96c33beae9a
                         ',
                      'xmpp.example.com␣example.org␣D60000229F'
                   )
    = '37
       c69b1cf07a3f67c04a5ef5902fa5114f2c76fe4a2686482ba5b89323075643
       '
```

9. The Authoritative Server compares this value to the key contained in the verification requests and informs the Originating Server of the result, in our example a success:

```
<db:verify
  to='xmpp.example.com'
  from='example.org'
  id='D60000229F'
  type='valid'/>
```

4 Security Considerations

This document introduces no security considerations or concerns above and beyond those discussed in RFC 3920 and XEP-0220.

5 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [6].

[6] 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/>.

6 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar [7].

7 Acknowledgements

Thanks to Ian Paterson and Matthias Wimmer for their feedback.

[7] 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/>.