XEP-0300: Use of Cryptographic Hash Functions in XMPP

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

Matthew Wild
mailto:mwild1@gmail.com
xmpp:me@matthewwild.co.uk

Kevin Smith
mailto:kevin@kismith.co.uk
xmpp:kevin@doomsong.co.uk

Tobias Markmann
mailto:tobias.markmann@isode.com
xmpp:tm@ayena.de

2018-02-14
Version 0.5.3

Status	Type	Short Name
Experimental	Standards Track	N/A

This document provides recommendations for the use of cryptographic hash functions in XMPP protocol extensions.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Various XMPP extensions make use of cryptographic hash functions, but they do so in different ways (e.g., some define XML elements and some define XML attributes) and often mandate support for different algorithms. The lack of a consistent approach to the use of cryptographic hash functions in XMPP extensions can lead to interoperability problems and security vulnerabilities. Therefore, this document recommends a common approach and XML element that can be re-used in any XMPP protocol extension.

2 Requirements

This extension is designed to meet the following criteria:

Agility It is absolutely necessary to support more secure cryptographic hash functions as they become available, and to stop supporting less secure functions as they are deprecated.

Security This document needs to be regularly maintained and revisited so that XMPP protocols are using the most up-to-date security technologies.

Reusability The extension needs to be reusable in any XMPP protocol.

3 XML Format

This document defines a new XML element that can be used in any XMPP protocol extension. An example follows.

```
<hash xmlns='urn:xmpp:hashes:2' algo='sha-256'>2XarmwTlNxDAMkvymloX3S5
    +VbylNrJt/l5QyPa+YoU=</hash>
```

An XMPP protocol can include more than one instance of the <hash/> element, as long as each one has a different value for the 'algo' attribute:

```
<hash xmlns='urn:xmpp:hashes:2' algo='sha-1'>2
    AfMGH8O7UNPTvUVAM9aK13mpCY=</hash>
<hash xmlns='urn:xmpp:hashes:2' algo='sha-256'>2XarmwTlNxDAMkvymloX3S5
    +VbylNrJt/l5QyPa+YoU=</hash>
```

In certain scenarios it makes sense to communicate the hash algorithm that is used prior to the calculation of the hash value.

```
<hash-used xmlns='urn:xmpp:hashes:2' algo='sha-256'/>
```

The value of the 'algo' attribute MUST be one of the values from the IANA Hash Function Textual Names Registry [1] maintained by the Internet Assigned Numbers Authority (IANA) [2], or one of the values defined in the following table.

Hash Function Name	Reference
"sha3-256"	FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions <http://dx.doi.org/10.6028/NIST.FIPS.202>.
"sha3-512"	FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions <http://dx.doi.org/10.6028/NIST.FIPS.202>.
"blake2b-256"	RFC 7693 RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC) <http://tools.ietf.org/html/rfc7693>.
"blake2b-512"	RFC 7693 RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC) <http://tools.ietf.org/html/rfc7693>.

The digest produced by the used hash algorithm is included as the XML character data of the <hash/> element after being encoded using Base64 as specified in Section 4 of RFC 4648 [3]. Thus the character data MUST conform to the base64Binary datatype [4] as defined in XML Schema Part 2 [5]. The Base64 output MUST NOT include whitespace and MUST set padding bits to zero.

4 Hash Functions

4.1 MD2

The MD2 algorithm is not used in any XMPP protocols and has been deprecated by the IETF (see RFC 6149 [6]).

[1] IANA registry of Hash Function Textual Names <http://www.iana.org/assignments/hash-function-text-names>.

[2] 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/>.

[3] RFC 4648: The Base16, Base32, and Base64 Data Encodings <http://tools.ietf.org/html/rfc4648>.

[4] See <http://www.w3.org/TR/xmlschema-2/#base64Binary>.

[5] XML Schema Part 2: Datatypes <http://www.w3.org/TR/xmlschema11-2/>.

[6] RFC 6149: MD2 to Historic Status <http://tools.ietf.org/html/rfc6149>.

4.2 MD4

The MD4 algorithm is not used in any XMPP protocols and has been deprecated by the IETF (see RFC 6150 [7]).

4.3 MD5

The MD5 algorithm was commonly used in earlier generations of Internet technologies. As explained in RFC 6151 [8], the MD5 algorithm "is no longer acceptable where collision resistance is required" (such as in digital signatures) and "new protocol designs should not employ HMAC-MD5" either.
The currently known best attack against the pre-image resistance property of the MD5 algorithm is slightly better than the generic attack and was released 2009 [9].
The primary use of MD5 in XMPP protocols is SI File Transfer (XEP-0096) [10], which will be obsoleted by Jingle File Transfer (XEP-0234) [11].

4.4 SHA-0

The SHA-0 algorithm was developed by the U.S. National Security Agency and first published in 1993. It was never widely deployed and is not used in any XMPP protocols.

4.5 SHA-1

The SHA-1 algorithm was developed by the U.S. National Security Agency and first published in 1995 to fix problems with SHA-0. The SHA-1 algorithm is currently the most widely-deployed hash function. As described in RFC 4270 [12] in 2005, attacks have been found against the collision resistance property of SHA-1. RFC 6194 [13] notes that as of 2011 no published results indicate improvement upon those attacks. In addition, RFC 6194 notes that "[t]here are no known pre-image or second pre-image attacks that are specific to the full round SHA-1 algorithm". Furthermore, there is no indication that attacks on SHA-1 can be extended to HMAC-SHA-1. Nevertheless, the U.S. National Institute of Standards and Technology (NIST) has recommended that SHA-1 not be used for generating digital signatures after December 31, 2010.

[7] RFC 6150: MD4 to Historic Status <http://tools.ietf.org/html/rfc6150>.
[8] RFC 6151: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms <http://tools.ietf.org/html/rfc6151>.
[9] Yu Sasaki and Kazumaro Aoki, "Finding preimages in full MD5 faster than exhaustive search" <https://doi.org/10.1007/978-3-642-01001-9_8>.
[10] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[11] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[12] RFC 4270: Attacks on Cryptographic Hashes in Internet Protocols <http://tools.ietf.org/html/rfc4270>.
[13] RFC 6194: Updated Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms <http://tools.ietf.org/html/rfc6194>.

In fall 2015 the SHA-1 collision cost has been estimated between 75K$ to 120K$ [14].
The SHA-1 algorithm is used in a number of XMPP protocols. See Analysis of Existing XMPP Extensions for details.

4.6 SHA-2

The SHA-2 family of algorithms (SHA-224, SHA-256, SHA-384, and SHA-512) was developed by the U.S. National Security Agency and first published in 2001. Because SHA-2 is somewhat similar to SHA-1, it is thought that the security flaws with SHA-1 described above could be extended to SHA-2 (although no such attacks have yet been found on the full-round SHA-2 algorithms).

4.7 SHA-3

The SHA-3 family of algorithms (SHA3-224, SHA3-256, SHA3-384, and SHA3-512) is based on the Keccak algortihm developed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche, and was pubished by NIST on August 5, 2015 in FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions [15] after a public hash function competition.

4.8 BLAKE2

The BLAKE2 family of algorithms was designed by Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. It is described in RFC 7693 [16] and is designed to be highly secure and run well on both software and hardware platforms.

5 Algorithm Recommendations

Support for version 1 of the 'urn:xmpp:hashes' namespace implies the following:

Algorithm	Digest Size	Support
MD2	128 bits	MUST NOT
MD4	128 bits	MUST NOT
MD5	128 bit	MUST NOT

[14] The SHAppening: freestart collisions for SHA-1 <https://sites.google.com/site/itstheshappening/>.
[15] FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions <http://dx.doi.org/10.6028/NIST.FIPS.202>.
[16] RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC) <http://tools.ietf.org/html/rfc7693>.

Algorithm	Digest Size	Support
SHA-1	160 bits	SHOULD NOT
SHA-256	256 bits	MUST
SHA-512	512 bits	SHOULD
SHA3-256	256 bits	MUST
SHA3-512	512 bits	SHOULD
BLAKE2b256	256 bits	MUST
BLAKE2b512	512 bits	SHOULD

These recommendations ought to be reviewed yearly by the XMPP Council [17].

6 Determining Support

If an entity supports the protocol defined herein, it MUST report that by including a Service Discovery (XEP-0030) [18] feature of "urn:xmpp:hashes:2" in response to disco#info requests, along with one service discovery feature for each algorithm it supports:

Listing 1: Service discovery information request

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

Listing 2: Service discovery information response

```
<iq from='juliet@capulet.lit/balcony'
    id='uw72g176'
    to='romeo@montague.lit/orchard'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <feature var='urn:xmpp:hashes:2'/>
    <feature var='urn:xmpp:hash-function-text-names:sha-256'/>
    <feature var='urn:xmpp:hash-function-text-names:sha3-256'/>
  </query>
</iq>
```

[17] The XMPP Council is a technical steering committee, authorized by the XSF Board of Directors and elected by XSF members, that approves of new XMPP Extensions Protocols and oversees the XSF's standards process. For further information, see <https://xmpp.org/about/xmpp-standards-foundation#council>.

[18] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

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) [19]. However, if an application has not received entity capabilities information from an entity, it SHOULD use explicit service discovery instead.

7 Recommendations for New XMPP Extensions

The XSF is strongly encouraged to incorporate hash agility into new XMPP extensions that it develops by mandating re-use of the protocol defined in this specification (instead of hash elements or attributes specific to each extension).

8 Analysis of Existing XMPP Extensions

As mentioned, several existing XMPP extensions make use of the SHA-1 algorithm. This section analyzes those extensions. The final subsection provides recommendations.

8.1 XEP-0065

Both SOCKS5 Bytestreams (XEP-0065) [20] and Jingle SOCKS5 Bytestreams Transport Method (XEP-0260) [21] use SHA-1 to hash the Stream ID, Requester's JID, and Target's JID, and this hash can be communicated via the 'dstaddr' attribute. Although this usage is not security-critical, currently it has no agility to specify newer algorithms. Because the hash is communicated by means of an attribute, it cannot directly use the extension defined in this specification.

8.2 XEP-0084

In User Avatar (XEP-0084) [22], the Publish-Subscribe (XEP-0060) [23] ItemId for the metadata node is the SHA-1 hash of the image data for the "image/png" media type. There is no hash agility for this usage. Although attacks against the collision resistance property could potentially result in confusion over the avatar for a user, the fact that avatars cannot be uploaded without authentication as the node owner or authorization as a node publisher reduces the practicality of attacks. In addition, XEP-0084 ought to be updated to specify that avatars must not be compared across JIDs.

[19] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[20] XEP-0065: SOCKS5 Bytestreams <https://xmpp.org/extensions/xep-0065.html>.
[21] XEP-0260: Jingle SOCKS5 Bytestreams Transport Method <https://xmpp.org/extensions/xep-0260.html>.
[22] XEP-0084: User Avatar <https://xmpp.org/extensions/xep-0084.html>.
[23] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

8.3 XEP-0115

Entity Capabilities (XEP-0115) [24] typically uses SHA-1 to compute the verification string, however hash agility is supported by use of the 'hash' attribute. Because the hash is communicated by means of an attribute, it cannot directly use the extension defined in this specification.

8.4 XEP-0124

BOSH (XEP-0124) [25] uses SHA-1 to generate the key sequence used to secure sessions that are not protected via SSL/TLS. Because these keys are ephemeral, it is unlikely that an attacker could reproduce or poison the key sequence quickly enough to successfully attack the session. However, attackers can be discouraged more significantly by protecting sessions with SSL/TLS (indeed, it is unclear how widely the key sequence feature is implemented). That said, this use of SHA-1 in BOSH does not support hash agility.

8.5 XEP-0153

vCard-Based Avatars (XEP-0153) [26] is historical but still widely used. Probably it is more valuable to modify XEP-0084 so that it supports hash agility.

8.6 XEP-0174

Link-Local Messaging (XEP-0174) [27] uses SHA-1 to hash the avatar image (i.e., the "phsh" field) advertised in the DNS TXT record for a user, mirroring the usage from XEP-0115. The "hash" field can be used to specify alternative hash algorithms, and thus supports hash agility. However, in practice it is likely that only SHA-1 is implemented. Because the hash is represented in a DNS TXT record, it cannot directly use the extension defined in this specification.

8.7 XEP-0231

Bits of Binary (XEP-0231) [28] supports hash agility through the structure of values for the 'cid' attribute, but does not mandate support for any particular algorithm.

[24] XEP-0115: Entity Capabilities <https://xmpp.org/extensions/xep-0115.html>.
[25] XEP-0124: Bidirectional-streams Over Synchronous HTTP <https://xmpp.org/extensions/xep-0124.html>.
[26] XEP-0153: vCard-Based Avatars <https://xmpp.org/extensions/xep-0153.html>.
[27] XEP-0174: Link-Local Messaging <https://xmpp.org/extensions/xep-0174.html>.
[28] XEP-0231: Bits of Binary <https://xmpp.org/extensions/xep-0231.html>.

8.8 XEP-0234

Jingle File Transfer (XEP-0234) [29] supports hash agility in its application format to allow to verify integrity of transferred files. It does not mandate support for any particular algorithm.

8.9 Recommendations

Of the foregoing, the use in XEP-0115 has the most significant security implications. However, there are other security issues with XEP-0115 that make it likely to be replaced in a more wholesale fashion. Although it would be desirable for all XMPP extensions that use cryptographic hashes to incorporate hash agility, realistically this is difficult to achieve after the fact. For now, the XSF is encouraged to focus on new protocols (e.g., XEP-0234 and a replacement for XEP-0115 if there is consensus to work on the latter) rather than spending effort on migrating its existing uses of SHA-1 to the SHA-2 family of algorithms, and to the SHA-3 family when available. Naturally, these priorities might change if XMPP technologies experience significant attacks on existing extensions that use SHA-1.

9 Security Considerations

This entire document discusses security.

10 IANA Considerations

This document requires no interaction with the IANA. However, it reuses entries from the relevant IANA registry.

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

This specification defines the following XML namespace:

- urn:xmpp:hashes:2

The XMPP Registrar [30] shall include the foregoing namespace in its registry at <https://xmpp.org/registrar/namespaces.html>, as governed by XMPP Registrar Func-

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

[30] 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/>.

tion (XEP-0053) [31].

11.2 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.

11.3 Service Discovery Features

An entity SHOULD provide one service discovery feature for each algorithm it supports. Ideally these features would be of the form "urn:iana:hash-function-text-names:foo" (where "foo" is the name of an algorithm registered with the IANA); however there is no urn:iana namespace at present. Until there is, we use features of the form "urn:xmpp:hash-function-text-names:foo" instead. Therefore the registry submission is as follows.

```
<var>
  <name>urn:xmpp:hash-function-text-names:md5</name>
  <desc>Support for the MD5 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>

<var>
  <name>urn:xmpp:hash-function-text-names:sha-1</name>
  <desc>Support for the SHA-1 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>

<var>
  <name>urn:xmpp:hash-function-text-names:sha-224</name>
  <desc>Support for the SHA-224 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha-256</name>
  <desc>Support for the SHA-256 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha-384</name>
  <desc>Support for the SHA-384 hashing algorithm</desc>
  <doc>XEP-0300</doc>
```

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

```
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha-512</name>
  <desc>Support for the SHA-512 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>

<var>
  <name>urn:xmpp:hash-function-text-names:sha3-224</name>
  <desc>Support for the SHA3-224 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha3-256</name>
  <desc>Support for the SHA3-256 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha3-384</name>
  <desc>Support for the SHA3-384 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:sha3-512</name>
  <desc>Support for the SHA3-512 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>

<var>
  <name>urn:xmpp:hash-function-text-names:id-blake2b160</name>
  <desc>Support for the BLAKE2b-160 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:id-blake2b256</name>
  <desc>Support for the BLAKE2b-256 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:id-blake2b384</name>
  <desc>Support for the BLAKE2b-384 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
<var>
  <name>urn:xmpp:hash-function-text-names:id-blake2b512</name>
  <desc>Support for the BLAKE2b-512 hashing algorithm</desc>
  <doc>XEP-0300</doc>
</var>
```

12 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:hashes:2'
    xmlns='urn:xmpp:hashes:2'
    elementFormDefault='qualified'>

  <xs:element name='hash'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:base64Binary'>
          <xs:attribute name='algo' type='xs:NCName' use='required'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='hash-used'>
    <xs:complexType>
      <xs:extension base='empty'>
        <xs:attribute name='algo' type='xs:NCName' use='required'/>
      </xs:extension>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

13 Acknowledgements

Thanks to Dave Cridland, Waqas Hussain, Glenn Maynard, Remko Tronçon, Paul Schaub, Christian Schudt, and Florian Schmaus for their input.