XEP-0414: Cryptographic Hash Function Recommendations for XMPP

Jonas Schäfer
mailto:jonas@wielicki.name
xmpp:jonas@wielicki.name

2020-05-23
Version 0.4.0

Status	Type	Short Name
Deferred	Informational	hashrecs

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

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

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.
This specification provides recommendations from the XMPP council as to which cryptographic hash functions should and should not be used by XMPP entities.

1.1 Relationship with Specifications

This recommendation does not specify the hash algorithms themselves; it merely refers to existing algorithms.
Use of Cryptographic Hash Functions in XMPP (XEP-0300) [1] (which historically has contained the recommendations in this specification) describes a common wire-format to be used to transport hash function values in XMPP.

2 Requirements

This recommendation should meet the following goals:

- Provide clear guidance on which hash functions should be supported by an XMPP entity at any point in time.
- Recommend both a set of well-supported functions as MUST and a set of future functions as SHOULD to allow the ecosystem to transit to newer functions.

This specification is *not* meant to override recommendations or requirements laid out by other specifications. Other specifications can however defer their recommendations or requirements to this specification.

3 Use Cases

A specification which makes use of cryptographic hash functions (such as Jingle File Transfer (XEP-0234) [2] or Entity Capabilities 2.0 (XEP-0390) [3]) can refer to this specification instead of making recommendations on hash functions on their own.
If a protocol specification defers its decision on hash functions to this document, it should support transporting multiple hashes at the same time (preferably using Use of Cryptographic

[1] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[2] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.
[3] XEP-0390: Entity Capabilities 2.0 <https://xmpp.org/extensions/xep-0390.html>.

Hash Functions in XMPP (XEP-0300) [4]).
By default, when an entity receives multiple hash function values for the same input, it SHOULD either (a) use all hash values or (b) the hash value of the algroithm with the most security confidence for verification purposes.

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 [5]).

4.2 MD4

The MD4 algorithm is not used in any XMPP protocols and has been deprecated by the IETF (see RFC 6150 [6]).

4.3 MD5

The MD5 algorithm was commonly used in earlier generations of Internet technologies. As explained in RFC 6151 [7], 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 [8].
The primary use of MD5 in XMPP protocols is SI File Transfer (XEP-0096) [9], which will be obsoleted by Jingle File Transfer (XEP-0234) [10].

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] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[5] RFC 6149: MD2 to Historic Status <http://tools.ietf.org/html/rfc6149>.
[6] RFC 6150: MD4 to Historic Status <http://tools.ietf.org/html/rfc6150>.
[7] RFC 6151: Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms <http://tools.ietf.org/html/rfc6151>.
[8] 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>.
[9] XEP-0096: SI File Transfer <https://xmpp.org/extensions/xep-0096.html>.
[10] XEP-0234: Jingle File Transfer <https://xmpp.org/extensions/xep-0234.html>.

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 [11] in 2005, attacks have been found against the collision resistance property of SHA-1. RFC 6194 [12] 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.
In fall 2015 the SHA-1 collision cost has been estimated between 75K$ to 120K$ [13].

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 [14] 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 [15] and is designed

[11] RFC 4270: Attacks on Cryptographic Hashes in Internet Protocols <http://tools.ietf.org/html/rfc4270>.
[12] RFC 6194: Updated Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms <http://tools.ietf.org/html/rfc6194>.
[13] The SHAppening: freestart collisions for SHA-1 <https://sites.google.com/site/itstheshappening/>.
[14] FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions <http://dx.doi.org/10.6028/NIST.FIPS.202>.
[15] RFC 7693: The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC) <http://tools.ietf.org/html/rfc7693>.

to be highly secure and run well on both software and hardware platforms.

5 Algorithm Recommendations

The current recommendations are as follows:

Algorithm	Digest Size	Support
MD2	128 bits	MUST NOT
MD4	128 bits	MUST NOT
MD5	128 bit	MUST NOT
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	SHOULD
BLAKE2b512	512 bits	MUST

These recommendations ought to be reviewed yearly by the XMPP Council [16].

6 Acknowledgements

Thanks to the authors and involved people in Use of Cryptographic Hash Functions in XMPP (XEP-0300) [17]; This specification is a mostly verbatim excerpt of a Use of Cryptographic Hash Functions in XMPP (XEP-0300) [18] version 0.5.3.

[16] 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>.
[17] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.
[18] XEP-0300: Use of Cryptographic Hash Functions in XMPP <https://xmpp.org/extensions/xep-0300.html>.