Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 13, 2019

J. Benet
Protocol Labs
M. Sporny
Digital Bazaar

August 12, 2018

The Multihash Data Format
draft-multiformats-multihash-00

Abstract

Cryptographic hash functions often have multiple output sizes and encodings. This variability makes it difficult for applications to examine a series of bytes and determine which hash function produced them. Multihash is a universal data format for encoding outputs from hash functions. It is useful to write applications that can simultaneously support different hash function outputs as well as upgrade their use of hashes over time; Multihash is intended to address these needs.

Feedback

This specification is a joint work product of Protocol Labs [1], the W3C Digital Verification Community Group [2], and the W3C Credentials Community Group [3]. Feedback related to this specification should logged in the issue tracker [4] or be sent to public-credentials@w3.org [5]. .

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 13, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

Multihash is particularly important in systems which depend on cryptographically secure hash functions. Attacks may break the cryptographic properties of secure hash functions. These cryptographic breaks are particularly painful in large tool

ecosystems, where tools may have made assumptions about hash values, such as function and digest size. Upgrading becomes a nightmare, as all tools which make those assumptions would have to be upgraded to use the new hash function and new hash digest length. Tools may face serious interoperability problems or error-prone special casing.

How many programs out there assume a git hash is a SHA-1 hash?

How many scripts assume the hash value digest is exactly 160 bits?

How many tools will break when these values change?

How many programs will fail silently when these values change?

This is precisely why Multihash was created. It was designed for seamlessly upgrading systems that depend on cryptographic hashes.

When using Multihash, a system warns the consumers of its hash values that these may have to be upgraded in case of a break. Even though the system may still only use a single hash function at a time, the use of multihash makes it clear to applications that hash values may use different hash functions or be longer in the future. Tooling, applications, and scripts can avoid making assumptions about the length, and read it from the multihash value instead. This way, the vast majority of tooling - which may not do any checking of hashes - would not have to be upgraded at all. This vastly simplifies the upgrade process, avoiding the waste of hundreds or thousands of software engineering hours, deep frustrations, and high blood pressure.

2. The Multihash Fields

A multihash follows the TLV (type-length-value) pattern and consists of several fields composed of a combination of unsigned variable length integers and byte information.

2.1. Multihash Core Data Types

The following section details the core data types used by the Multihash data format.

2.1.1. unsigned variable integer

A data type that enables one to express an unsigned integer of variable length.

When encoding an unsigned variable integer, the unsigned integer is serialized seven bits at a time, starting with the least significant

bits. The most significant bit in each output byte intecates if there is a continuation byte. It is not possible to express a signed integer with this data type.

Value	Encoding (bits)
1	00000001
127	01111111
128	10000000 00000001
255	11111111 00000001
300	10101100 00000010
16384	10000000 10000000 00000001

Table 1: Examples of Unsigned Variable Integers

Implementations MUST restrict the size of the varint to a max of nine bytes (63 bits). In order to avoid memory attacks on the encoding, the aforementioned practical maximum length of of nine bytes is used. There is no theoretical limit, and future specs can grow this number if it is truly necessary to have code or length values larger than 2^31.

2.2. Multihash Fields

A multihash follows the TLV (type-length-value) pattern.

2.2.1. Hash Function Type

The hash function type is an unsigned variable integer [6] identifying the hash function. The possible values for this field are provided in The Multihash Algorithms Registry [7].

2.2.2. Digest Length

The digest length is an unsigned variable integer [8] counting the length of the digest in bytes.

2.2.3. Digest Value

The digest value is the hash function digest with a length of exactly what is specified in the digest length, which is specified in bytes.

2.3. A Multihash Example

For example, the following is an expression of a SHA2-256 hash in hexadecimal notation (spaces added for readability purposes):

```
12 20 41dd7b6443542e75701aa98a0c235951a28a0d851b11564d20022ab11d2589a8
```

The first byte (0x12) specifies the SHA2-256 hash function. The second byte (0x20) specifies the length of the hash, which is 32 bytes. The rest of the data specifies the value of the output of the hash function.

3. References

3.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7693] Saarinen, M-J., Ed. and J-P. Aumasson, "The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)", RFC 7693, DOI 10.17487/RFC7693, November 2015, <https://www.rfc-editor.org/info/rfc7693>.

3.2. Informative References

[RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, DOI 10.17487/RFC6150, March 2011, <https://www.rfc-editor.org/info/rfc6150>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

3.3. URIs

[1] https://protocol.ai/

[2] https://w3c-dvcg.github.io/

[3] https://w3c-ccg.github.io/

[4] https://github.com/w3c-dvcg/multihash/issues

[5] mailto:public-credentials@w3.org

[6] #cdt-uvi

[7] #mh-registry

[8] #cdt-uvi

[9] http://www.iana.org/assignments/multihash-algorithms

[10] https://github.com/multiformats/multihash/blob/master/hashtable.csv

Appendix A. Security Considerations

There are a number of security considerations to take into account when implementing or utilizing this specification. TBD

Appendix B. Test Values

The multihash examples are chosen to show different hash functions and different hash digest lengths at play. The input test data for all of the examples in this section is:

B.1. SHA-1

The fields for this multihash are - hashing function: sha1 (0x11), length: 20 (0x14), digest: 8a173fd3e32c0fa78b90fe42d305f202244e2739

B.2. SHA-256

The fields for this multihash are - hashing function: sha2-256 (0x12), length: 32 (0x20), digest: 41dd7b6443542e75701aa98a0c235951a28a0d851b11564d20022ab11d2589a8

B.3. SHA-512/256

The fields for this multihash are - hashing function: sha2-512 (0x13), length: 32 (0x20), digest: 52eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4

B.4. SHA-512

The fields for this multihash are - hashing function: sha2-512 (0x13), length: 64 (0x40), digest: 52eb4dd19f1ec522859e12d89706156570f8fbab1824870bc6f8c7d235eef5f4c2cbbafd365f96fb12b1d98a0334870c2ce90355da25e6a1108a6e17c4aaebb0

B.5. blake2b512

The fields for this multihash are - hashing function: blake2b-512 (0xb240), length: 64 (0x40), digest: d91ae0cb0e48022053ab0f8f0dc78d28593d0f1c13ae39c9b169c136a779f21a0496337b6f776a73c1742805c1cc15e792ddb3c92ee1fe300389456ef3dc97e2

B.6. blake2b256

The fields for this multihash are - hashing function: blake2b-256 (0xb220), length: 32 (0x20), digest: 7d0a1371550f3306532ff44520b649f8be05b72674e46fc24468ff74323ab030

B.7. blake2s256

The fields for this multihash are - hashing function: blake2s-256 (0xb260), length: 32 (0x20), digest: a96953281f3fd944a3206219fad61a40b992611b7580f1fa091935db3f7ca13d

B.8. blake2s128

The fields for this multihash are - hashing function: blake2s-128 (0xb250), length: 16 (0x10), digest: 0a4ec6f1629e49262d7093e2f82a3278

Appendix C. Acknowledgements

The editors would like to thank the following individuals for feedback on and implementations of the specification (in alphabetical order).

Appendix D. IANA Considerations

D.1. The Multihash Algorithms Registry

The following initial entries should be added to the Multihash Algorithms Registry to be created and maintained at (the suggested URI) http://www.iana.org/assignments/multihash-algorithms [9]:

Codec	Identifier (hex)	Status	Specification

identity	0x00	active	Unknown
md4	0xd4	deprecated	RFC 6150 [RFC6150]
md5	0xd5	deprecated	RFC 6151 [RFC6151]
sha1	0x11	active	RFC 6234 [RFC6234]
sha2-256	0x12	active	RFC 6234 [RFC6234]
sha2-512	0x13	active	RFC 6234 [RFC6234]
dbl-sha2-256	0x56	active	Unknown
sha3-224	0x17	active	Unknown
sha3-256	0x16	active	Unknown
sha3-384	0x15	active	Unknown
sha3-512	0x14	active	Unknown
shake-128	0x18	active	Unknown
shake-256	0x19	active	Unknown
keccak-224	0x1A	active	Unknown
keccak-256	0x1B	active	Unknown
keccak-384	0x1C	active	Unknown
keccak-512	0x1D	active	Unknown
murmur3-128	0x22	active	Unknown
murmur3-32	0x23	active	Unknown
blake2b-8	0xb201	active	RFC 7693 [RFC7693]
blake2b-16	0xb202	active	RFC 7693 [RFC7693]
blake2b-24	0xb203	active	RFC 7693 [RFC7693]
blake2b-32	0xb204	active	RFC 7693 [RFC7693]
blake2b-40	0xb205	active	RFC 7693 [RFC7693]
blake2b-48	0xb206	active	RFC 7693 [RFC7693]
blake2b-56	0xb207	active	RFC 7693 [RFC7693]
blake2b-64	0xb208	active	RFC 7693 [RFC7693]
blake2b-72	0xb209	active	RFC 7693 [RFC7693]
blake2b-80	0xb20a	active	RFC 7693 [RFC7693]
blake2b-88	0xb20b	active	RFC 7693 [RFC7693]
blake2b-96	0xb20c	active	RFC 7693 [RFC7693]
blake2b-104	0xb20d	active	RFC 7693 [RFC7693]
blake2b-112	0xb20e	active	RFC 7693 [RFC7693]
blake2b-120	0xb20f	active	RFC 7693 [RFC7693]
blake2b-128	0xb210	active	RFC 7693 [RFC7693]
blake2b-136	0xb211	active	RFC 7693 [RFC7693]
blake2b-144	0xb212	active	RFC 7693 [RFC7693]
blake2b-152	0xb213	active	RFC 7693 [RFC7693]
blake2b-160	0xb214	active	RFC 7693 [RFC7693]
blake2b-168	0xb215	active	RFC 7693 [RFC7693]
blake2b-176	0xb216	active	RFC 7693 [RFC7693]
blake2b-184	0xb217	active	RFC 7693 [RFC7693]
blake2b-192	0xb218	active	RFC 7693 [RFC7693]
blake2b-200	0xb219	active	RFC 7693 [RFC7693]
blake2b-208	0xb21a	active	RFC 7693 [RFC7693]
blake2b-216	0xb21b	active	RFC 7693 [RFC7693]
blake2b-224	0xb21c	active	RFC 7693 [RFC7693]
blake2b-232	0xb21d	active	RFC 7693 [RFC7693]

blake2b-240	0xb21e	active	RFC 7693 [RFC7693]
blake2b-248	0xb21f	active	RFC 7693 [RFC7693]
blake2b-256	0xb220	active	RFC 7693 [RFC7693]
blake2b-264	0xb221	active	RFC 7693 [RFC7693]
blake2b-272	0xb222	active	RFC 7693 [RFC7693]
blake2b-280	0xb223	active	RFC 7693 [RFC7693]
blake2b-288	0xb224	active	RFC 7693 [RFC7693]
blake2b-296	0xb225	active	RFC 7693 [RFC7693]
blake2b-304	0xb226	active	RFC 7693 [RFC7693]
blake2b-312	0xb227	active	RFC 7693 [RFC7693]
blake2b-320	0xb228	active	RFC 7693 [RFC7693]
blake2b-328	0xb229	active	RFC 7693 [RFC7693]
blake2b-336	0xb22a	active	RFC 7693 [RFC7693]
blake2b-344	0xb22b	active	RFC 7693 [RFC7693]
blake2b-352	0xb22c	active	RFC 7693 [RFC7693]
blake2b-360	0xb22d	active	RFC 7693 [RFC7693]
blake2b-368	0xb22e	active	RFC 7693 [RFC7693]
blake2b-376	0xb22f	active	RFC 7693 [RFC7693]
blake2b-384	0xb230	active	RFC 7693 [RFC7693]
blake2b-392	0xb231	active	RFC 7693 [RFC7693]
blake2b-400	0xb232	active	RFC 7693 [RFC7693]
blake2b-408	0xb233	active	RFC 7693 [RFC7693]
blake2b-416	0xb234	active	RFC 7693 [RFC7693]
blake2b-424	0xb235	active	RFC 7693 [RFC7693]
blake2b-432	0xb236	active	RFC 7693 [RFC7693]
blake2b-440	0xb237	active	RFC 7693 [RFC7693]
blake2b-448	0xb238	active	RFC 7693 [RFC7693]
blake2b-456	0xb239	active	RFC 7693 [RFC7693]
blake2b-464	0xb23a	active	RFC 7693 [RFC7693]
blake2b-472	0xb23b	active	RFC 7693 [RFC7693]
blake2b-480	0xb23c	active	RFC 7693 [RFC7693]
blake2b-488	0xb23d	active	RFC 7693 [RFC7693]
blake2b-496	0xb23e	active	RFC 7693 [RFC7693]
blake2b-504	0xb23f	active	RFC 7693 [RFC7693]
blake2b-512	0xb240	active	RFC 7693 [RFC7693]
blake2s-8	0xb241	active	RFC 7693 [RFC7693]
blake2s-16	0xb242	active	RFC 7693 [RFC7693]
blake2s-24	0xb243	active	RFC 7693 [RFC7693]
blake2s-32	0xb244	active	RFC 7693 [RFC7693]
blake2s-40	0xb245	active	RFC 7693 [RFC7693]
blake2s-48	0xb246	active	RFC 7693 [RFC7693]
blake2s-56	0xb247	active	RFC 7693 [RFC7693]
blake2s-64	0xb248	active	RFC 7693 [RFC7693]
blake2s-72	0xb249	active	RFC 7693 [RFC7693]
blake2s-80	0xb24a	active	RFC 7693 [RFC7693]
blake2s-88	0xb24b	active	RFC 7693 [RFC7693]
blake2s-96	0xb24c	active	RFC 7693 [RFC7693]
blake2s-104	0xb24d	active	RFC 7693 [RFC7693]

blake2s-112	0xb24e	active	RFC 7693 [RFC7693]
blake2s-120	0xb24f	active	RFC 7693 [RFC7693]
blake2s-128	0xb250	active	RFC 7693 [RFC7693]
blake2s-136	0xb251	active	RFC 7693 [RFC7693]
blake2s-144	0xb252	active	RFC 7693 [RFC7693]
blake2s-152	0xb253	active	RFC 7693 [RFC7693]
blake2s-160	0xb254	active	RFC 7693 [RFC7693]
blake2s-168	0xb255	active	RFC 7693 [RFC7693]
blake2s-176	0xb256	active	RFC 7693 [RFC7693]
blake2s-184	0xb257	active	RFC 7693 [RFC7693]
blake2s-192	0xb258	active	RFC 7693 [RFC7693]
blake2s-200	0xb259	active	RFC 7693 [RFC7693]
blake2s-208	0xb25a	active	RFC 7693 [RFC7693]
blake2s-216	0xb25b	active	RFC 7693 [RFC7693]
blake2s-224	0xb25c	active	RFC 7693 [RFC7693]
blake2s-232	0xb25d	active	RFC 7693 [RFC7693]
blake2s-240	0xb25e	active	RFC 7693 [RFC7693]
blake2s-248	0xb25f	active	RFC 7693 [RFC7693]
blake2s-256	0xb260	active	RFC 7693 [RFC7693]

Table 2: Multihash Algorithms Registry

NOTE: The most up to date place for developers to find the table above is https://github.com/multiformats/multihash/blob/master/hashtable.csv [10].

Authors' Addresses

Juan Benet
Protocol Labs
548 Market Street, #51207
San Francisco, CA 94104
US

Phone: +1 619 957 7606
Email: juan@protocol.ai
URI: http://juan.benet.ai/

Manu Sporny
Digital Bazaar
203 Roanoke Street W.
Blacksburg, VA 24060
US

Phone: +1 540 961 4469
Email: msporny@digitalbazaar.com
URI: http://manu.sporny.org/