Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2017

P. Hallam-Baker
Comodo Group Inc.

September 19, 2016

Uniform Data Fingerprint (UDF)
draft-hallambaker-udf-04

Abstract

This document describes means of generating Uniform Data Fingerprint (UDF) values and their presentation as text sequences and as URIs.

Cryptographic digests provide a means of uniquely identifying static data without the need for a registration authority. A fingerprint is a form of presenting a cryptographic digest that makes it suitable for use in applications where human readability is required. The UDF fingerprint format improves over existing formats through the introduction of a compact algorithm identifier affording an intentionally limited choice of digest algorithm and the inclusion of an IANA registered MIME Content-Type identifier within the scope of the digest input to allow the use of a single fingerprint format in multiple application domains.

Alternative means of rendering fingerprint values are considered including machine-readable codes, word and image lists.

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 http://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 March 23, 2017.

Table of Contents

1. Definitions

Cryptographic Digest Function

Digest

Fingerprint

Hash

Presentation

Fingerprint Strengthening

Fingerprint Work Hardening

Work Factor

Content-Type

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Introduction

The use of cryptographic digest functions to produce identifiers is well established as a means of generating a unique identifier for fixed data without the need for a registration authority.

While the use of fingerprints of public keys was popularized by PGP, they are employed in many other applications including OpenPGP, SSH, BitCoin and PKIX.

A cryptographic digest is a particular form of hash function that has the properties:

It is easy to compute the digest value for any given message

It is infeasible to generate a message from its digest value

It is infeasible to modify a message without changing the digest value

It is infeasible to find two different messages with the same digest value.

If these properties are met, the only way that two data objects that map to the same digest value is by random chance. If the number of possible digest values is sufficiently large (i.e. is a sufficiently large number of bits in length), this chance is reduced to an arbitrarily infinitesimal probability. Such values are described as being probabilistically unique.

A fingerprint is a representation of a cryptographic digest value optimized for purposes of verification and in some cases data entry.

2.1. Algorithm Identifier

Although a secure cryptographic digest algorithm has properties that make it ideal for certain types of identifier use, several cryptographic digest algorithms have found widespread use, some of which have been demonstrated to be insecure.

For example the MD5 message digest algorithm [RFC1321], was widely used in IETF protocols until it was demonstrated to be vulnerable to collision attacks [TBS].

The secure use of a fingerprint scheme therefore requires the digest algorithm to either be fixed or otherwise determined by the fingerprint value itself. Otherwise an attacker may be able to use a weak, broken digest algorithm to generate a data object matching a fingerprint value generated using a strong digest algorithm.

2.2. Content Type Identifier

A secure cryptographic digest algorithm provides a unique digest value that is probabilistically unique for a particular byte sequence but does not fix the context in which a byte sequence is interpreted. While such ambiguity may be tolerated in a fingerprint format designed for a single specific field of use, it is not acceptable in a general purpose format.

For example, the SSH and OpenPGP applications both make use of fingerprints as identifiers for the public keys used but using different digest algorithms and data formats for representing the public key data. While no such vulnerability has been demonstrated to date, it is certainly conceivable that a crafty attacker might construct an SSH key in such a fashion that OpenPGP interprets the

data in an insecure fashion. If the number of applications making use of fingerprint format that permits such substitutions is sufficiently large, the probability of a semantic substitution vulnerability being possible becomes unacceptably large.

A simple control that defeats such attacks is to incorporate a content type identifier within the scope of the data input to the hash function.

2.3. Representation

The representation of a fingerprint is the format in which it is presented to either an application or the user.

Base32 encoding is used to produce the preferred text representation of a UDF fingerprint. This encoding uses only the letters of the Latin alphabet with numbers chosen to minimize the risk of ambiguity between numbers and letters (2, 3, 4, 5, 6 and 7).

To enhance readability and improve data entry, characters are grouped into groups of five.

2.4. Truncation

Different applications of fingerprints demand different tradeoffs between compactness of the representation and the number of significant bits. A larger the number of significant bits reduces the risk of collision but at a cost to convenience.

Modern cryptographic digest functions such as SHA-2 produce output values of at least 256 bits in length. This is considerably larger than most uses of fingerprints require and certainly greater than can be represented in human readable form on a business card.

Since a strong cryptographic digest function produces an output value in which every bit in the input value affects every bit in the output value with equal probability, it follows that truncating the digest value to produce a finger print is at least as strong as any other mechanism if digest algorithm used is strong.

Using truncation to reduce the precision of the digest function has the advantage that a lower precision fingerprint of some data content is always a prefix of a higher prefix of the same content. This allows higher precision fingerprints to be converted to a lower precision without the need for special tools.

3. Encoding

A UDF fingerprint for a given data object is generated by calculating the Binary Fingerprint Value for the given data object and type identifier, truncating it to obtain the desired degree of precision and then converting the truncated value to a representation.

3.1. Binary Fingerprint Value

The binary encoding of a fingerprint is calculated using the formula:

Fingerprint = <<Version-ID> + H (<<Content-ID> + ?:? + H(<<Data>))

Where

H(x) is the cryptographic digest function
<<Version-ID> is the fingerprint version and algorithm identifier.
<<Content-ID> is the MIME Content-Type of the data.
<<Data> is the binary data.

The use of the nested hash function permits a fingerprint to be taken of data for which a digest value is already known without the need to calculate a new digest over the data.

The inclusion of a MIME content type prevents message substitution attacks in which one content type is substituted for another.

3.1.1. Version ID

Two digest algorithm identifiers are specified in this document:

SHA-2-512 = 96

SHA-3-512 = 144

These algorithm identifiers have been carefully chosen so that the first character in a SHA-2-512 fingerprint will always be 'M' and the first character in a SHA-3-512 fingerprint will always be 'S'. These provide mnemonics for 'Merkle-Damgard' and 'Sponge' respectively.

3.2. Truncation

The Binary Fingerprint Value is truncated to an integer multiple of 25 bits regardless of the intended output presentation.

The output of the hash function is truncated to a sequence of n bits by first selecting the first n/8 bytes of the output function. If n is an integer multiple of 8, no additional bits are required and this

is the result. Otherwise the remaining bits are taken from the most significant bits of the next byte and any unused bits set to 0.

For example, to truncate the byte sequence [a0, b1, c2, d3, e4] to 25 bits. 25/8 = 3 bytes with 1 bit remaining, the first three bytes of the truncated sequence is [a0, b1, c2] and the final byte is e4 AND 80 = 80 which we add to the previous result to obtain the final truncated sequence of [a0, b1, c2, 80]

3.3. Base32 Representation

A modified version of Base32 [RFC4648] encoding is used to present the fingerprint in text form grouping the output text into groups of five characters separated by a dash '-'. This representation improves the accuracy of both data entry and verification.

3.4. URI Representation

Any UDF fingerprint MAY be encoded as a URI by prefixing the Base32 text representation of the fingerprint with the string 'udf:'

3.5. Examples

In the following examples, <Content-ID> is the UTF8 encoding of the string "text/plain" and is the UTF8 encoding of the string "UDF Data Value"

Data = 55 44 46 20 44 61 74 61 20 56 61 6c 75 65

3.5.1. Using SHA-2-512 Digest

```
H( <Data> ) =
    48 da 47 cc  ab fe a4 5c  76 61 d3 21  ba 34 3e 58
    10 87 2a 03  b4 02 9d ab  84 7c ce d2  22 b6 9c ab
    02 38 d4 e9  1e 2f 6b 36  a0 9e ed 11  09 8a ea ac
    99 d9 e0 bd  ea 47 93 15  bd 7a e9 e1  2e ad c4 15
H(H( <Data> ) + Content-ID>) =
    45 e0 59 e0  39 34 ea b7  f6 5d 83 b2  d8 f9 b1 6d
    2a 6b 08 63  d9 3c c1 02  86 7b 83 49  f2 d9 f0 8f
    fe 07 87 30  c7 c9 05 74  ac a1 38 2b  b3 14 4d c6
    39 f9 8c 12  c0 4a 3e b5  05 0b 3e 67  df 52 4b 57
```

Text Presentation (100 bit)MB2GK-6DUF5-YGYYL-JNY5E

Text Presentation (125 bit)MB2GK-6DUF5-YGYYL-JNY5E-RWSHZ

Text Presentation (150bit)MB2GK-6DUF5-YGYYL-JNY5E-RWSHZ-SV75J

Text Presentation (250bit)MB2GK-6DUF5-YGYYL-JNY5E-RWSHZ-SV75J-C4OZQ-5GIN2-GQ7FQ-EEHFI

3.5.2. Using SHA-3-512 Digest

[This data intentionally omitted pending publication of the final SHA-3 standards document]

3.6. Key Improvement

3.7. Work Hardening

4. Content Types

4.1. PKIX keyInfo

4.2. OpenPGP Key

5. Additional UDF Renderings

By default, a UDF fingerprint is rendered in the Base32 encoding described in this document. Additional renderings MAY be employed to facilitate entry and/or verification of fingerprint values.

5.1. Machine Readable Rendering

The use of a machine-readable rendering such as a QR Code allows a UDF value to be input directly using a smartphone or other device equipped with a camera.

A QR code fixed to a network capable device might contain the fingerprint of a machine readable description of the device.

5.2. Word Lists

The use of a Word List to encode fingerprint values was introduced by Patrick Juola and Philip Zimmerman for the PGPfone application. The PGP Word List is designed to facilitate exchange and verification of fingerprint values in a voice application. To minimize the risk of misinterpretation, two word lists of 256 values each are used to encode alternative fingerprint bytes. The compact size of the lists used allowed the compilers to curate them so as to maximize the phonetic distance of the words selected.

The PGP Word List is designed to achieve a balance between ease of entry and verification. Applications where only verification is required may be better served by a much larger word list, permitting shorter fingerprint encodings.

For example, a word list with 16384 entries permits 14 bits of the fingerprint to be encoded at once, 65536 entries permits 16. These encodings allow a 125 bit fingerprint to be encoded in 9 and 8 words respectively.

5.3. Image List

An image list is used in the same manner as a word list affording rapid visual verification of a fingerprint value. For obvious reasons, this approach is not generally suited to data entry.

6. Security Considerations

6.1. Precision

6.2. Use of Truncated Digests

7. IANA Considerations

[This will be extended later]

7.1. URI Registration

[Here a URI registration for the udf: scheme]

7.2. Content Type Registration

[PKIX KeyInfo]

[PGP Key Packet]

7.3. Version Registry

96 = SHA-2-512

144 = SHA-3-512

8. Normative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006.

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com