Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 7, 2012

S. Farrell
Trinity College Dublin
D. Kutscher
NEC
C. Dannewitz
University of Paderborn
B. Ohlman
A. Keranen
Ericsson
P. Hallam-Baker
Comodo Group Inc.

April 5, 2012

Naming Things with Hashes
draft-farrell-decade-ni-02

Abstract

This document defines a set of ways to identify a thing using the output from a hash function, specifying URI, URL, binary and human "speakable" formats for these names. The various formats are designed to support, but not require, a strong link to the referenced object such that the referenced object may be authenticated to the same degree as the reference to it.

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 October 7, 2012.

Table of Contents

1. Introduction

Names or identifiers are used in various protocols for identifying resources. In many scenarios those names or identifiers contain values that are hash function outputs. However, different deployments have chosen various different ways to include hash function outputs in such names or identifiers. There is a benefit in specifying a standard way to include hash function outputs in such names or identifiers so we do that here.

Hash function outputs can be used to ensure uniqueness in terms of mapping URIs [RFC3986] to a specific resource, or to make URIs hard to guess for security reasons. Since, there is no standard way to interpret those strings, today in general only the creator of the URI knows how to use the hash function output. Other protocols, such as application layer protocols for accessing "smart objects" in constrained environments also require more compact (e.g., binary) forms of such identifiers, while in other situations people may have to input such values or talk about them, e.g., in a voice call.

As another example, protocols for accessing in-network storage servers need a way to identify stored resources uniquely and in a location-independent way so that replicas on different servers can be accessed by the same name. Also, such applications may require verifying that a resource representation that has been obtained actually corresponds to the name that was used to request the resource, i.e., verifying the name-content binding.

Similarly, in the context of information-centric networking [ref.netinf-design] [ref.ccn] and elsewhere there is value in being able to compare a presented resource against the URI that was dereferenced in order to access that resource. If a cryptographically-strong comparison function can be used then this allows for many forms of in-network storage, without requiring as much trust in the infrastructure used to present the resource. The outputs of hash functions can be used in this manner, if presented in a standard way.

Additional applications might include creating references from web pages delivered over HTTP/TLS; DNS resource records signed using DNSSEC or Data values embedded in certificates, Certificate Revocation Lists (CRLs), or other signed data objects.

The new URI scheme defined here allows for the use of a query-string, similar to how query-strings are used in HTTP URLs. A companion specification [niexts] describes specific values that can be used in such query strings for various purposes and other extensions to this basic format specification.

The "ni" URI scheme defined here is very similar to the "magnet link" informally defined in various other protocols. [magnet]

In addition to the URI form we also define a ".well-known" URL equivalent, and a way to include a hash as a fragment of an HTTP URL, as well as a binary format for use in protocols that require more compact names and a human-speakable text form that could be used, e.g. for reading out (parts of) the name over a voice connection.

Not all uses of these names require use of the full hash output - truncated hashes can be safely used in some environments. For this reason, we define a new IANA registry for hash functions to be used with this specification so as not to mix strong and weak hash algorithms in other protocol registries.

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].

Syntax definitions in this memo are specified according to ABNF [RFC4648].

[[Comments are included in double-square brackets, like this.]]

2. Basics

This section contains basic considerations common to all formats.

When verifying whether two names refer to same object, an implementation MUST only consider the digest algorithm identifier and the digest value, i.e., it MUST NOT consider the authority field from a URI or any parameters and MUST consider two hashes identical regardless of encoding, if they encode the same binary value, and are the same length.

The sha-256 algorithm as specified in [RFC4055] is mandatory to implement, that is, implementations MUST be able to generate/send and to accept/process names based on a sha-256 hash. However implementations MAY support additional hash algorithms and MAY use those for specific names, for example in a constrained environment where sha-256 is non-optimal or where truncated names are needed to fit into corresponding protocols (when a higher collision probability can be tolerated).

Truncated hashes MAY be supported if needed. When a hash value is truncated the name MUST indicate this. Therefore we use different hash algorithm strings for these, such as sha-256-32 for a 32-bit

truncation of a sha-256 output. (Note that a 32-bit truncated hash is essentially useless for security but might be useful for naming.)

When a hash value is truncated to N bits the left-most or most significant in network byte order N bits from the binary representation of the hash value MUST be used as the truncated value. An example of a 128-bit hash output truncated to 32 bits is shown in Figure 1.

```
          128-bit hash: 0x265357902fe1b7e2a04b897c6025d7a2
  32-bit truncated hash: 0x26535790
```

Figure 1: Example of Truncated Hash

When the input to the hash algorithm is a public key value, as may be used by various security protocols, the hash SHOULD be calculated over the public key in an X.509 SubjectPublicKeyInfo structure (Section 4.1 of [RFC5280]). This input has been chosen primarily for compatibility with DANE [I-D.ietf-dane-protocol], but also includes any relevant public key parameters in the hash input, which is sometimes necessary for security reasons. Note also that this does not force use of X.509 or full compliance with [RFC5280] since formatting any public key as a SubjectPublicKeyInfo is relatively straightforward and well supported by libraries.

Any of the formats defined below can be used to represent the resulting name for a public key.

3. ni URI Format

An ni URI consists of the following components:

Scheme Name [Required] The scheme name is 'ni'.

Colon and Slashes [Required] The literal "://"

Authority [Optional] The optional authority component may assist applications in accessing the object named by an ni URI. Note that while the ni names with and without an authority differ syntactically, both names will almost always refer to the same object.

One slash [Required] The literal "/"

Digest Algorithm [Required] The name of the digest algorithm, as specified in the IANA registry defined in Section 9.1 below.

Separator [Required] The literal ";"

Digest Value [Required] The digest value encoded in the specified encoding.

Query Parameter separator [Optional] '?' The query parameter separator acts a separator between the digest value and the query parameters (if specified).

Query Parameters [Optional] A tag=value list of optional query parameters as are used with HTTP URLs.

It is OPTIONAL for implementations to check the integrity of the URI/resource mapping when sending, receiving or processing "ni" URIs.

The digest value MUST be encoded using base64url [RFC4648] encoding.

The query segment of an URI is NOT hierarchical. Thus escape encoding of slash '/' characters is NOT required. Since application code often attempts to enforce such encoding, decoders MUST recognize the use of URI escape encoding. Section 3.4 of [RFC3986] states that "The characters slash ("/") and question mark ("?") may represent data within the query component."

Consequently no special escaping mechanism is required for the query parameter portion of ni URIs. URI escaping is however frequently imposed automatically by scripting environments. Thus to ensure interoperability, implementations SHOULD NOT generate URIs that employ URI character escaping, and implementations MUST NOT reject any URIs that employ URI character escaping.

The Named Information URI has the following syntax:

```
niname ="ni://" [ authority ] "/" algval [ "?" query ]
algval = alg ";" val
alg = 1*CHAR
val = 1*CHAR
```

Figure 2: ni Name syntax

The "authority" and "query" types are as in the URI specification. [RFC3986]

The "val" field MUST contain the output of applying the hash function ("alg") to its defined input, which defaults to the object bytes that

are expected to be returned when the URI is dereferenced.

4. .well-known URL Format

We define a mapping between URIs following the ni URI scheme and HTTP or HTTPS URLs that makes use of the .well-known URI [RFC5785] by defining an "ni" suffix (see Section 9).

The HTTP(S) mapping MAY be used in any context where clients without support for ni URIs are needed without loss of interoperability or functionality.

For an ni name of the form "ni://n-authority/alg;val?query-string" the corresponding HTTP(S) URL produced by this mapping is "http://h-authority/.well-known/ni/alg/val?query-string", where "h-authority" is derived as follows: If the ni name has a specified authority (i.e., the n-authority is non-empty) then the h-authority MUST have the same value. If the ni name has no authority specified (i.e. the n-authority string is empty), a h-authority value MAY be derived from the application context. For example, if the mapping is being done in the context of a web page then the origin [RFC6454] for that web site can be used. Of course, there are in general no guarantees that the object named by the ni URI will be available at the corresponding HTTP(S) URL. But in the case that any data is returned, the retreiver can determine whether or not it is content that matches the ni URI.

If an application is presented with a HTTP(S) URL with "/.well-known/ni/" as the start of its pathname component, then the reverse mapping to an ni URI either including or excluding the authority might produce an ni URI that is meaningful, but there is no guarantee that this will be the case.

When mapping from a ni URI to a .well-known URL, an implementation will have to decide between choosing an "http" or "https" URL. If the object referenced does in fact match the hash in the URL, then there is arguably no need for additional data integrity, if the ni URI or .well-known URL was received "securely." However TLS also provides confidentiality, so there may still be reasons to use the "https" URL scheme even in this case. In general however, whether to use "http" or "https" is something that needs to be decided by the application.

5. URL Fragment Format

Some applications may benefit from using hashes in existing HTTP URLs

or other URLs. To do this one simply uses the "algval" production from the ni name scheme ABNF which may be included in the pathname component of HTTP URLs. In such cases there is nothing present in the URL that ensures that a client can depend on compliance with this specification, so clients MUST NOT assume that any URL with a pathname component that matches the "algval" production was in fact produced as a result of this specification. That URL might or might not be related to this specification, only the context will tell.

6. Binary Format

When a more space-efficient version of the name is needed, we can use a binary format. The binary format name consists of two fields: a header and the hash value. The header field defines how the identifier has been created and the hash value contains a (possibly truncated) result of a one-way hash over the whatever is being identified by the hash value. The binary representation of the name is shown in Figure 3.

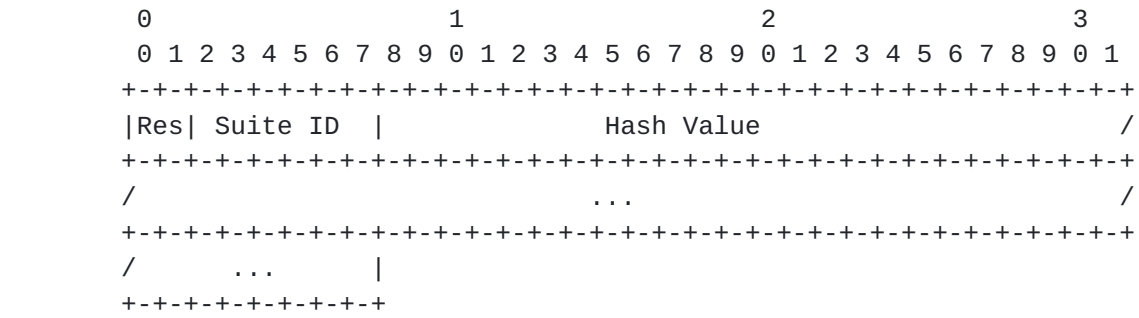

Figure 3: Binary Name Format

The Res field is a reserved 2-bit field for future use and MUST be set to zero for this specification.

The hash algorithm and truncation length are specified by the Suite ID. For maintaining efficient encoding for the binary presentation, only a few hash algorithms and truncation lengths are supported. See Section Section 9.3 for details.

Note that a hash value that is truncated to 120 bits will result in the overall name being a 128-bit value which may be useful with certain use-cases.

7. Human-readable Format

Sometimes the name may need to be used in a format that is easy for humans to read and possibly communicate, for example, over the phone. For this purpose, the following more verbose but less ambiguous (when spoken) format is defined.

As with the ni URI format, the fields are separated by a semi-colon (;) character. The first field is a hash algorithm string, as in the ni URI format. Then the hash value is encoded using base32 encoding [RFC4648] and lower-case alphabets. Since the length of the hash value is known from the hash algorithm string, padding characters are not needed and SHOULD NOT be used when the result of the base32 encoding is used in the hash value field. When decoding the base32-encoded string, only first N bits of the result, depending on the truncation as indicated by the hash algorithm string, MUST be used.

The hash value is OPTIONALLY followed by a checksum. The checksum MUST be calculated as a crc16 over the parts preceding the checksum, i.e., the algorithm string, the first delimiter, (";") the hash value, and the second delimiter (also ";") is also included in the input to the crc16 calculation. The result of crc16 is encoded like the hash value with base32 encoding and lower-case alphabets (only the first 4 characters of the base32 encoded result are used to exclude padding).

The crc16 MUST use the CRC-CCITT polynomial: x^16 + x^12 + x^5 + 1.

[[CCITT crc16 needs a proper reference]]

```
humanname = algval [ ";" checksum ]
algval = alg ";" val
alg = 1*CHAR
val = 1*CHAR
checksum = 1*CHAR
```

Figure 4: Human-readable syntax

8. Examples

The following digest URI specifies a reference to the text "Hello World !" using the SHA-2 algorithm with 256-bit output and no authority field:

ni:///sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc

And the same example shown with an authority would be:

ni://example.com/sha-256;B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc

The following HTTP URL represents a mapping from the previous ni name based on the algorithm outlined above.

http://example.com/.well-known/ni/sha-256/
B_K97zTtFuOhug27fke4_Zgc4Myz4b_lZNgsQjy6fkc

Given the SubjectPublicKeyInfo in Figure 5 we derive the names shown in Figure 6 for this value.

```
0000000 8230 2201 0d30 0906 862a 8648 0df7 0101
0000020 0501 0300 0182 000f 8230 0a01 8202 0101
0000040 a200 835f 9bda f1d9 3a7a 6736 fdba 945a
0000060 cf0e d516 555a 5e3a 03d4 65b1 6d8e a3cf
0000100 dbb7 e7a4 0fcc c652 357d c41d c268 7bbd
0000120 db9d 0ae4 10d7 f9cd 2053 0dee 56d7 5b6e
0000140 ae7a 5f2c 0a83 3c19 5872 d696 e886 e60e
0000160 eb94 f25c 3e90 a8f3 888a b656 36cd 7638
0000200 9722 6bb1 9c3c f307 974f a108 29bc 9b38
0000220 0681 742b 3860 937a 392f 12be 0934 0b6e
0000240 1057 a3b7 f27b eec6 c1d6 ece5 c5ae 839c
0000260 f414 586b dee2 fff2 77c9 e307 4cf3 cf97
0000300 281a 389e b3a1 4193 a175 76a4 4d3f d778
0000320 d644 e31a e2ce c55d 4c78 31b5 2e22 4bc7
0000340 6f8c 7856 a15c c0c4 ca1d b9e5 d744 90e9
0000360 bc9c b0ee b1a2 dadc a06d f60f 1ead 122c
0000400 a7a2 6066 363e 91d4 c241 e7f2 3969 9d2c
0000420 dfd2 a3b5 9544 7c48 6487 dd89 05bf ee01
0000440 02dd 0103 0100

0000000 2653 5790 2fe1 b7e2 a04b 897c 6025 d7a2
0000020 8753 b67e f42f 5a4d 0019 3025 97ed e4ff
```

Figure 5: A SubjectPublicKeyInfo used in examples and its sha-256 hash

```
+--------------------------------------------------------------------+
| URI:                                                               |
| ni:///sha-256;UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q          |
+--------------------------------------------------------------------+
| .well-known URL (split over 2 lines):                              |
| http://tcd.ie/.well-known/ni/sha256/                               |
| UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q                        |
+--------------------------------------------------------------------+
| URL Fragment:                                                      |
| sha-256;UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q                |
+--------------------------------------------------------------------+
| Binary name (ASCII hex encoded) with 120-bit truncated hash value  |
| which is Suite ID 0x03:                                            |
| 0326 5357 902f e1b7 e2a0 4b89 7c60 25d7                            |
+--------------------------------------------------------------------+
| Human-readable form of a name for this key (truncated to 120 bits  |
| in length) with checksum:                                          |
| sha-256-120;ezjvpebp4g36ficlrf6gajox;eokv                          |
+--------------------------------------------------------------------+
| Human readable form of a name for this key (truncated to 32 bits   |
| in length) with checksum:                                          |
| sha-256-32;ezjvpea;csdh                                            |
+--------------------------------------------------------------------+
```

Figure 6: Example Names

9. IANA Considerations

9.1. Assignment of Network Information (ni) URI Scheme

The procedures for registration of a URI scheme are specified in RFC 4395 [RFC4395]. The following is the proposed assignment template.

URI scheme name: ni

Status: Permanent

URI scheme syntax. See Section 3

URI scheme semantics. See Section 3

Encoding considerations. See Section 3

Applications/protocols that use this URI scheme name: General applicability with initial use cases provided by WEBSEC and DECADE

Interoperability considerations: Defined here.

Security considerations: See Section 10

Contact: stephen.farrell@cs.tcd.ie

Author/Change controller: IETF

References: As specified in this document

9.2. Assignment of Well Known URI prefix ni

The procedures for registration of a Well Known URI entry are specified in RFC 5785 [RFC5785]. The following is the proposed assignment template.

URI suffix: ni

Change controller: IETF

Specification document(s): This document

Related information: None

9.3. Binary Suite IDs

IANA is requested to create a new registry for hash algorithms as used in the name formats specified here. This registry has four fields, the binary suite ID, the hash algorithm name string, the truncation length and the underlying algorithm reference. Future assignments are to be made through expert review. [RFC5226]. Initial values are specified below.

ID	Hash name String	Value length	Reference
0	Reserved		
1	sha-256	256 bits	[RFC4055]
2	sha-256-128	128 bits	[RFC4055]
3	sha-256-120	120 bits	[RFC4055]
4	sha-256-96	96 bits	[RFC4055]
5	sha-256-64	64 bits	[RFC4055]
6	sha-256-32	32 bits	[RFC4055]
32	Reserved		

Figure 7: Suite Identifiers

The Suite ID value 32 is reserved for compatibility with ORCHIDs [RFC4843]. The referenced hash algorithm matching to the Suite ID, truncated to the length indicated, according to the description given

in Section 2, MUST be used for generating the hash.

[[Do we need sha-1 here? Its been asked for, but in a new standards track spec is dodgy...]]

10. Security Considerations

No secret information is required to generate or verify a name of the form described here. Therefore a name like this can only provide evidence for the integrity for the referenced object and the proof of integrity provided is only as good as the proof of integrity for the name from which we started. In other words, the hash value can provide a name-data integrity binding between the name and the bytes returned when the name is de-referenced using some protocol.

Disclosure of a name value does not necessarily entail disclosure of the referenced object but may enable an attacker to determine the contents of the referenced object by reference to a search engine or other data repository or, for a highly formatted object with little variation, by simply guessing the value and checking if the digest value matches. So the fact that these names contain hashes does not necessarily protect the confidentiality of the object that was input to the hash.

The integrity of the referenced content would be compromised if a weak hash function were used. So don't use those. SHA-256 is currently our preferred hash algorithm which is why we've only added SHA-256 based suites to the initial IANA registry.

If a truncated hash value is used, certain security properties might be affected. In general a hash algorithm is designed to produce sufficient bits to prevent a 'birthday attack' collision occurring. To ensure that the difficulty of discovering two pieces of content that result in the same digest with a work factor O(2^x) by brute force requires a digest length of 2x. Many security applications only require protection against a 2nd pre-image attack which only requires a digest length of x to achieve the same work factor. Basically, the shorter the hash value used, the less security benefit you can possibly get.

11. Acknowledgements

This work has been supported by the EU FP7 project SAIL. The authors would like to thank SAIL participants to our naming discussions, especially Jean-Francois Peltier, for their input.

The authors would also like to thank Bob Moskowitz, Tero Kivinen, Zach Shelby, Carsten Bormann, David McGrew, Eric Rescorla, and Tobias Heer for their comments and input to the document.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, June 2005.

[RFC4395] Hansen, T., Hardie, T., and L. Masinter, "Guidelines and Registration Procedures for New URI Schemes", BCP 35, RFC 4395, February 2006.

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

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

12.2. Informative References

[I-D.ietf-dane-protocol] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Protocol for Transport Layer Security (TLS)", draft-ietf-dane-protocol-18 (work in progress), March 2012.

[RFC4843] Nikander, P., Laganier, J., and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers

(ORCHID)", RFC 4843, April 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, December 2011.

[magnet] Wikipedia article, "Magnet URI Scheme", April 2012, <http://en.wikipedia.org/wiki/Magnet_link>.

[niexts] Hallam-Baker, P., Stradling, R., Farrell, S., Kutscher, C., and B. Ohlman, "The Network Information (ni) URI Scheme: Parameters", draft-hallambaker-decade-ni-params-00 (work in progress), October 2011.

[ref.ccn] Jacobsen, K, D, F, H, and L, "Networking Named Content", CoNEXT 2009 , December 2009.

[ref.netinf-design] Ahlgren, D'Ambrosio, Dannewitz, Marchisio, Marsh, Ohlman, Pentikousis, Rembarz, Strandberg, and Vercellone, "Design Considerations for a Network of Information", Re-Arch 2008 Workshop , December 2008.

Authors' Addresses

Stephen Farrell
Trinity College Dublin
Dublin, 2
Ireland

Phone: +353-1-896-2354
Email: stephen.farrell@cs.tcd.ie

Dirk Kutscher
NEC
Kurfuersten-Anlage 36
Heidelberg,
Germany

Phone:
Email: kutscher@neclab.eu

Christian Dannewitz
University of Paderborn
Paderborn
Germany

Email: cdannewitz@upb.de

Borje Ohlman
Ericsson
Stockholm S-16480
Sweden

Email: Borje.Ohlman@ericsson.com

Ari Keranen
Ericsson
Jorvas 02420
Finland

Email: ari.keranen@ericsson.com

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com