Internet Engineering Task Force (IETF)
Request for Comments: 6920
Category: Standards Track
ISSN: 2070-1721

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 2013

Naming Things with Hashes

Abstract

This document defines a set of ways to identify a thing (a digital object in this case) using the output from a hash function. It specifies a new URI scheme for this purpose, a way to map these to HTTP URLs, and 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. The reason for this work is to standardise current uses of hash outputs in URLs and to support new information-centric applications and other uses of hash outputs in protocols.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Identifiers -- names or locators -- are used in various protocols to identify resources. In many scenarios, those identifiers contain values that are obtained from hash functions. Different deployments have chosen different ways to include the hash function outputs in their identifiers, resulting in interoperability problems.

This document defines the "Named Information" identifier, which provides a set of standard ways to use hash function outputs in names. We begin with a few example uses for various ways to include hash function output in a name, with the specifics defined later in this document. Figure 1 shows an example of the Named Information (ni) URI scheme that this document defines.

```
ni:///sha-256;UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q
```

Figure 1: Example ni URI

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. In yet other situations, people may have to speak such values, e.g., in a voice call (see Section 8.3), in order to confirm the presence or absence of a resource.

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 verification that a resource representation that has been obtained actually corresponds to the name that was used to request the resource, i.e., verifying the binding between the data and the name, which is here termed "name-data integrity".

Similarly, in the context of information-centric networking [NETINF-ARCHITECTURE] [CCN] and elsewhere, there is value in being able to compare a presented resource against the URI that was used 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 they are 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 Named Identifier can be represented in a number of ways: using the ni URI scheme that we specifically define for the name (which is very similar to the "magnet link" that is informally defined in other protocols [Magnet]), or using other mechanisms also defined herein. However it is represented, the Named Identifier *names* a resource, and the mechanism used to dereference the name and to *locate* the named resource needs to be known by the entity that dereferences it.

Media content-type, alternative locations for retrieval, and other additional information about a resource named using this scheme can be provided using a query string. "The Named Information (ni) URI Scheme: Optional Features" [DECPARAMS] describes specific values that can be used in such query strings for these various purposes and other extensions to this basic format specification.

In addition, we define a ".well-known" URL equivalent, a way to include a hash as a segment of an HTTP URL, 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 (truncated) 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 [RFC5234].

2. Hashes Are What Count

This section contains basic considerations related to how we use hash function outputs that are common to all formats.

When comparing two names of the form defined here, an implementation MUST only consider the digest algorithm and the digest value, i.e., it MUST NOT consider other fields defined below (such as an authority field from a URI or any parameters). Implementations MUST consider two hashes identical, regardless of encoding, if the decoded hashes are based on the same algorithm and have the same length and the same binary value. In that case, the two names can be treated as referring to the same thing.

The sha-256 algorithm as specified in [SHA-256] 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. When a hash value is truncated, the name MUST indicate this. Therefore, we use different hash algorithm strings in these cases, such as sha-256-32 for a 32-bit truncation of a sha-256 output. A 32-bit truncated hash is essentially useless for security in almost all cases but might be useful for naming. With current best practices [RFC3766], very few, if any, applications making use of names with less than 100-bit hashes will have useful security properties.

When a hash value is truncated to N bits, the leftmost N bits (that is, the most significant N bits in network byte order) 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 2.

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

Figure 2: 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 the DANE TSLA protocol [RFC6698] but also includes any relevant public key parameters in the hash input, which is sometimes necessary for security reasons. 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.

Other than in the aforementioned special case where public keys are used, we do not specify the hash function input here. Other specifications are expected to define this.

3. Named Information (ni) URI Format

A Named Information (ni) URI consists of the following nine components:

Scheme Name: The scheme name is 'ni'.

Colon and Slashes: The literal "://"

Authority: The optional authority component may assist applications in accessing the object named by an ni URI. There is no default value for the authority field. (See Section 3.2.2 of [RFC3986] for details.) While ni names with and without an authority differ syntactically from ni names with different authorities, all three refer to the same object if and only if the digest algorithm, length, and value are the same.

One slash: The literal "/"

Digest Algorithm: The name of the digest algorithm, as specified in the IANA registry defined in Section 9.4 below.

Separator: The literal ";"

Digest Value: The digest value MUST be encoded using the base64url [RFC4648] encoding, with no "=" padding characters.

Query Parameter separator '?': The query parameter separator acts as a separator between the digest value and the query parameters (if specified). For compatibility with Internationalized Resource Identifiers (IRIs), non-ASCII characters in the query part MUST be encoded as UTF-8, and the resulting octets MUST be percent-encoded (see [RFC3986], Section 2.1).

Query Parameters: A "tag=value" list of optional query parameters as are used with HTTP URLs [RFC2616] with a separator character '&' between each. For example, "foo=bar&baz=bat".

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

Escaping of characters follows the rules in RFC 3986. This means that percent-encoding is used to distinguish between reserved and unreserved functions of the same character in the same URI component.

As an example, an ampersand ('&') is used in the query part to separate attribute-value pairs; therefore, an ampersand in a value has to be escaped as '%26'. Note that the set of reserved characters differs for each component. As an example, a slash ('/') does not have any reserved function in a query part and therefore does not have to be escaped. However, it can still appear escaped as '%2f' or '%2F', and implementations have to be able to understand such escaped forms. Also note that any characters outside those allowed in the respective URI component have to be escaped.

The Named Information URI adapts the URI definition from the URI Generic Syntax [RFC3986]. We start with the base URI production:

```
URI = scheme ":" hier-part [ "?" query ] [ "#" fragment ]
    ; from RFC 3986
```

Figure 3: URI Syntax

Then, we adapt that for the Named Information URI:

```
NI-URI         = ni-scheme ":" ni-hier-part [ "?" query ]
    ; adapted from "URI" in RFC 3986
    ; query is from RFC 3986, Section 3.4
ni-scheme      = "ni"
ni-hier-part   = "//" [ authority ] "/" alg-val
    ; authority is from RFC 3986, Section 3.2
alg-val        = alg ";" val
    ; adapted from "hier-part" in RFC 3986
alg            = 1*unreserved
val            = 1*unreserved
    ; unreserved is from RFC 3986, Section 2.3
```

Figure 4: ni Name Syntax

The "val" field MUST contain the output of base64url encoding (with no "=" padding characters), the result 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.

Relative ni URIs can occur. In such cases, the algorithm in Section 5 of [RFC3986] applies. As an example, in Figure 5, the absolute URI for "this third document" is "ni://example.com/sha-256-128;...".

```
     <html>
      <head>
        <title>ni: relative URI test</title>
        <base href="ni://example.com">
      </head>

      <body>
        <p>Please check <a href="sha-256;f4OxZX...">this document</a>.
          and <a href="sha-256;UyaQV...">this other document</a>.
          and <a href="sha-256-128;...">this third document</a>.
        </p>
      </body>
     </html>
```

Figure 5: Example HTML with Relative ni URI

The authority field in an ni URI is not quite the same as that from an HTTP URL, even though the same values (e.g., DNS names) may be usefully used in both. For an ni URI, the authority does not control nearly as much of the structure of the "right-hand side" of the URI. With ni URIs we also define standard query string attributes and, of course, have a strictly defined way to include the hash value.

Internationalisation of strings within ni names is handled exactly as for http URIs -- see [RFC2616], Section 3.2.3.

3.1. Content Type Query String Attribute

The semantics of a digest being used to establish a secure reference from an authenticated source to an external source may be a function of associated metadata such as the Content Type. The Content Type "ct" parameter specifies the MIME Content Type of the associated data as defined in [RFC6838]. See Section 9.5 for the associated IANA registry for ni parameter names as shown in Figure 6. Implementations of this specification MUST support parsing the "ct=" query string attribute name.

```
ni:///sha-256-32;f4OxZQ?ct=text/plain
```

Figure 6: Example ni URI with Content Type

Protocols making use of ni URIs will need to specify how to verify name-data integrity for the MIME Content Types that they need to process and will need to take into account possible Content-Transfer-Encodings and other aspects of MIME encoding.

Implementations of this specification SHOULD support name-data integrity validation for at least the application/octet-stream Content Type, with no explicit Content-Transfer-Encoding (which is equivalent to binary). Additional Content Types and Content-Transfer-Encodings can of course also be supported, but are OPTIONAL. Note that the hash is calculated after the Content-Transfer-Encoding is removed so it is applied to the raw data.

If a) the user agent is sensitive to the Content Type and b) the ni name used has a "ct=" query string attribute and c) the object is retrieved (from a server) using a protocol that specifies a Content Type, then, if the two Content Types match, all is well. If, in this situation, the Content Types do not match, then the client SHOULD handle that situation as a potential security error. Content Type matching rules are defined in [RFC2045], Section 5.1.

4. .well-known URI

We define a mapping between URIs following the ni URI scheme and HTTP [RFC2616] or HTTPS [RFC2818] 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 with support for ni URIs are not available.

Since the .well-known name-space is not intended for general information retrieval, if an application dereferences a .well-known/ni URL via HTTP(S), then it will often receive a 3xx HTTP redirection response. A server responding to a request for a .well-known/ni URL will often therefore return a 3xx response, and a client sending such a request MUST be able to handle that, as should any fully compliant HTTP [RFC2616] client.

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, in general there are no guarantees that the object named by the ni URI will be available via the corresponding HTTP(S) URL. But in the case that any data is returned, the retriever can determine whether or not it is content that matches the ni URI.

If an application is presented with an 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. However, there is no guarantee that this will be the case.

When mapping from an 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 can still be reasons to use the "https" URL scheme even in this case. Additionally, web server policy such as [RFC6797] may dictate that data only be available over "https". In general, however, whether to use "http" or "https" is something that needs to be decided by the application.

5. URL Segment Format

Some applications may benefit from using hashes in existing HTTP URLs or other URLs. To do this, one simply uses the "alg-val" production from the ni name scheme ABNF, which may be included, for example, in the pathname, query string, or even fragment components of HTTP URLs [RFC2616]. 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 "alg-val" 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

If a more space-efficient version of the name is needed, the following binary format can be used. 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 whatever is being identified by the hash value. The binary format of a name is shown in Figure 7.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |Res| Suite ID  |              Hash Value                       /
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /                              ...                              /
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /      ...      |
    +-+-+-+-+-+-+-+-+
```

Figure 7: Binary Name Format

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

The hash algorithm and truncation length are specified by the Suite ID. For maintaining efficient encoding for the binary format, only a few hash algorithms and truncation lengths are supported. See Section 9.4 for details.

A hash value that is truncated to 120 bits will result in the overall name being a 128-bit value, which may be useful for protocols that can easily use 128-bit identifiers.

7. Human-Speakable (nih) URI Format

Sometimes a resource may need to be referred to via a name in a format that is easy for humans to read out and less likely to be ambiguous when heard. This is intended to be usable, for example, over the phone in order to confirm the (current or future) presence or absence of a resource. This "confirmation" use-case described further in Section 8.3 is the main current use-case for Named Information for Humans (nih) URIs. ("nih" also means "Not Invented Here", which is clearly false, and therefore worth including [RFC5513]. :-)

The ni URI format is not well-suited for this, as, for example, base64url uses both uppercase and lowercase, which can easily cause confusion. For this particular purpose ("speaking" the value of a hash output), the more verbose but less ambiguous (when spoken) nih URI scheme is defined.

The justification for nih being a URI scheme is that it can help a user agent for the speaker to better display the value or help a machine to better speak or recognise the value when spoken. We do not include the query string since there is no way to ensure that its value might be spoken unambiguously and similarly for the authority, where, e.g., some internationalised forms of domain name might not be easy to speak and comprehend easily. This leaves the hash value as the only part of the ni URI that we feel can be usefully included. But since speakers or listeners (or speech recognition) may err, we also include a checkdigit to catch common errors and allow for the inclusion of "-" separators to make nih URIs easier to read out.

Fields in nih URIs are separated by a semicolon (;) character. The first field is a hash algorithm string, as in the ni URI format. The hash value is represented using lowercase ASCII hex characters; for example, an octet with the decimal value 58 (0x3A) is encoded as '3a'. This is the same as base16 encoding as defined in RFC 4648 [RFC4648] except using lowercase letters. Separators ("-" characters) MAY be interspersed in the hash value in any way to make those easier to read, typically grouping four or six characters with a separator between.

The hash value MAY be followed by a semicolon ';' then a checkdigit. The checkdigit MUST be calculated using Luhn's mod N algorithm (with N=16) as defined in [ISOIEC7812] (see also [Luhn]). The input to the calculation is the ASCII hex-encoded hash value (i.e., "sepval" in the ABNF production below) but with all "-" separator characters first stripped out. This maps the ASCII hex so that '0'=0, ...'9'=9, 'a'=10, ...'f'=15. None of the other fields, nor any "-" separators, are input when calculating the checkdigit.

```
humanname  = "nih:" alg-sepval [ ";" checkdigit ]
alg-sepval = alg ";" sepval
sepval     = 1*(ahlc / "-")
ahlc       =  DIGIT / "a" / "b" / "c" / "d" / "e" / "f"
      ; DIGIT is defined in RFC 5234 and is 0-9
checkdigit = ahlc
```

Figure 8: Human-Speakable Syntax

For algorithms that have a Suite ID reserved (see Figure 11), the alg field MAY contain the ID value as an ASCII-encoded decimal number instead of the hash name string (for example, "3" instead of "sha-256-120"). Implementations MUST be able to match the decimal ID values for the algorithms and hash lengths that they support, even if they do not support the binary format.

There is no such thing as a relative nih URI.

8. Examples

8.1. Hello World!

The following ni URI is generated from the text "Hello World!" (12 characters without the quotes), using the sha-256 algorithm shown with and without an authority field:

```
ni:///sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

```
ni://example.com/sha-256;f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

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/
f4OxZX_x_FO5LcGBSKHWXfwtSx-j1ncoSt3SABJtkGk
```

8.2. Public Key Examples

Given the DER-encoded SubjectPublicKeyInfo in Figure 9, we derive the names shown in Figure 10 for this value.

```
0000000 30 82 01 22 30 0d 06 09 2a 86 48 86 f7 0d 01 01
0000020 01 05 00 03 82 01 0f 00 30 82 01 0a 02 82 01 01
0000040 00 a2 5f 83 da 9b d9 f1 7a 3a 36 67 ba fd 5a 94
0000060 0e cf 16 d5 5a 55 3a 5e d4 03 b1 65 8e 6d cf a3
0000100 b7 db a4 e7 cc 0f 52 c6 7d 35 1d c4 68 c2 bd 7b
0000120 9d db e4 0a d7 10 cd f9 53 20 ee 0d d7 56 6e 5b
0000140 7a ae 2c 5f 83 0a 19 3c 72 58 96 d6 86 e8 0e e6
0000160 94 eb 5c f2 90 3e f3 a8 8a 88 56 b6 cd 36 38 76
0000200 22 97 b1 6b 3c 9c 07 f3 4f 97 08 a1 bc 29 38 9b
0000220 81 06 2b 74 60 38 7a 93 2f 39 be 12 34 09 6e 0b
0000240 57 10 b7 a3 7b f2 c6 ee d6 c1 e5 ec ae c5 9c 83
0000260 14 f4 6b 58 e2 de f2 ff c9 77 07 e3 f3 4c 97 cf
0000300 1a 28 9e 38 a1 b3 93 41 75 a1 a4 76 3f 4d 78 d7
0000320 44 d6 1a e3 ce e2 5d c5 78 4c b5 31 22 2e c7 4b
0000340 8c 6f 56 78 5c a1 c4 c0 1d ca e5 b9 44 d7 e9 90
0000360 9c bc ee b0 a2 b1 dc da 6d a0 0f f6 ad 1e 2c 12
0000400 a2 a7 66 60 3e 36 d4 91 41 c2 f2 e7 69 39 2c 9d
0000420 d2 df b5 a3 44 95 48 7c 87 64 89 dd bf 05 01 ee
0000440 dd 02 03 01 00 01

0000000 53 26 90 57 e1 2f e2 b7 4b a0 7c 89 25 60 a2 d7
0000020 53 87 7e b6 2f f4 4d 5a 19 00 25 30 ed 97 ff e4
```

Figure 9: 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://example.com/.well-known/ni/sha256/                         |
   | UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q                       |
   +-------------------------------------------------------------------+
   | URL Segment:                                                      |
   | sha-256;UyaQV-Ev4rdLoHyJJWCi11OHfrYv9E1aGQAlMO2X_-Q               |
   +-------------------------------------------------------------------+
   | Binary name (ASCII hex encoded) with 120-bit truncated hash value |
   | which is Suite ID 0x03:                                           |
   | 0353 2690 57e1 2fe2 b74b a07c 8925 60a2                           |
   +-------------------------------------------------------------------+
   | Human-speakable form of a name for this key (truncated to 120 bits|
   | in length) with checkdigit:                                       |
   | nih:sha-256-120;5326-9057-e12f-e2b7-4ba0-7c89-2560-a2;f           |
   +-------------------------------------------------------------------+
   | Human-speakable form of a name for this key (truncated to 32 bits |
   | in length) with checkdigit and no "-" separators:                 |
   | nih:sha-256-32;53269057;b                                         |
   +-------------------------------------------------------------------+
   | Human-speakable form using decimal presentation of the            |
   | algorithm ID (sha-256-120) with checkdigit:                       |
   | nih:3;532690-57e12f-e2b74b-a07c89-2560a2;f                        |
   +-------------------------------------------------------------------+
```

Figure 10: Example Names

8.3. nih Usage Example

Alice has set up a server node with an RSA key pair. She uses an ni URI as the name for the public key that corresponds to the private key on that box. Alice's node might identify itself using that ni URI in some protocol.

Bob would like to believe that it's really Alice's node when his node interacts with the network and asks his friend Alice to tell him what public key she uses. Alice hits the "tell someone the name of the public key" button on her admin user interface and that displays the nih URI and says "tell this to your buddy". She phones Bob and reads the nih URI to him.

Bob types that in to his "manage known nodes" admin application (or lets that application listen to part of the call), which can regenerate the ni URI and store that or some equivalent. Then when Bob's node interacts with Alice's node, it can more safely accept a signature or encrypt data to Alice's node.

9. IANA Considerations

9.1. Assignment of ni URI Scheme

The procedures for registration of a URI scheme are specified in RFC 4395 [RFC4395]. The following assignment has been made.

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.

Interoperability considerations: Defined here.

Security considerations: See Section 10.

Contact: Stephen Farrell, stephen.farrell@cs.tcd.ie

Author/Change controller: IETF

References: As specified in this document

9.2. Assignment of nih URI Scheme

The procedures for registration of a URI scheme are specified in RFC 4395 [RFC4395]. The following assignment has been made.

URI scheme name: nih

Status: Permanent

URI scheme syntax: See Section 7.

URI scheme semantics: See Section 7.

Encoding considerations: See Section 7.

Applications/protocols that use this URI scheme name:

General applicability.

Interoperability considerations: Defined here.

Security considerations: See Section 10.

Contact: Stephen Farrell, stephen.farrell@cs.tcd.ie

Author/Change controller: IETF

References: As specified in this document

9.3. Assignment of .well-known 'ni' URI

The procedures for registration of a Well-Known URI entry are specified in RFC 5785 [RFC5785]. The following assignment has been made.

URI suffix: ni

Change controller: IETF

Specification document(s): This document

Related information: None

9.4. Creation of Named Information Hash Algorithm Registry

IANA has created a new registry for hash algorithms as used in the name formats specified here; it is called the "Named Information Hash Algorithm Registry". Future assignments are to be made through Expert Review [RFC5226]. This registry has five fields: the suite ID, the hash algorithm name string, the truncation length, the underlying algorithm reference, and a status field that indicates if the algorithm is current or deprecated and should no longer be used. The status field can have the value "current" or "deprecated". Other values are reserved for possible future definition.

If the status is "current", then that does not necessarily mean that the algorithm is "good" for any particular purpose, since the cryptographic strength requirements will be set by other applications or protocols.

A request to mark an entry as "deprecated" can be done by sending a mail to the Designated Expert. Before approving the request, the community MUST be consulted via a "call for comments" of at least two weeks by sending a mail to the IETF discussion list.

Initial values are specified below. The Designated Expert SHOULD generally approve additions that reference hash algorithms that are widely used in other IETF protocols. In addition, the Designated Expert SHOULD NOT accept additions where the underlying hash function (with no truncation) is considered weak for collisions. Part of the reasoning behind this last point is that inclusion of code for weak hash functions, e.g., the MD5 algorithm, can trigger costly false positives if code is audited for inclusion of obsolete ciphers. See [RFC6149], [RFC6150], and [RFC6151] for examples of some hash functions that are considered obsolete in this sense.

The suite ID field ("ID") can be empty or can have values between 0 and 63, inclusive. Because there are only 64 possible values, this field is OPTIONAL (leaving it empty if omitted). Where the binary format is not expected to be used for a given hash algorithm, this field SHOULD be omitted. If an entry is registered without a suite ID, the Designated Expert MAY allow for later allocation of a suite ID, if that appears warranted. The Designated Expert MAY consult the community via a "call for comments" by sending a mail to the IETF discussion list before allocating a suite ID.

ID	Hash Name String	Value Length	Reference	Status
0	Reserved			
1	sha-256	256 bits	[SHA-256]	current
2	sha-256-128	128 bits	[SHA-256]	current
3	sha-256-120	120 bits	[SHA-256]	current
4	sha-256-96	96 bits	[SHA-256]	current
5	sha-256-64	64 bits	[SHA-256]	current
6	sha-256-32	32 bits	[SHA-256]	current
32	Reserved			

Figure 11: Suite Identifiers

The Suite ID value 32 is reserved for compatibility with IPv6 addresses from the Special Purpose Address Registry [RFC4773], such as Overlay Routable Cryptographic Hash Identifiers (ORCHIDs) [RFC4843].

The referenced hash algorithm matching the Suite ID, truncated to the length indicated, according to the description given in Section 2, is used for generating the hash. The Designated Expert is responsible for ensuring that the document referenced for the hash algorithm meets the "specification required" rule.

9.5. Creation of Named Information Parameter Registry

IANA has created a new registry entitled "Named Information URI Parameter Definitions".

The policy for future assignments to the registry is Expert Review, and as for the ni Hash Algorithm Registry above, the Designated Expert is responsible for ensuring that the document referenced for the parameter definition meets the "specification required" rule.

The fields in this registry are the parameter name, a description, and a reference. The parameter name MUST be such that it is suitable for use as a query string parameter name in an ni URI. (See Section 3.)

The initial contents of the registry are:

Parameter	Meaning	Reference
ct	Content Type	[RFC6920]

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 of 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 dereferenced 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 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. SHA-256 is currently our preferred hash algorithm; this is why we've added only SHA-256-based suites to the initial IANA registry.

If a truncated hash value is used, certain security properties will be affected. In general, a hash algorithm is designed to produce sufficient bits to prevent a 'birthday attack' collision occurring. Ensuring 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 second 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.

An important thing to keep in mind is not to make the mistake of thinking two names are the same when they aren't. For example, a name with a 32-bit truncated sha-256 hash is not the same as a name with the full 256 bits of hash output, even if the hash value for one is a prefix of that for the other.

The reason for this is that if an application treats these as the same name, then that might open up a number of attacks. For example, if I publish an object with the full hash, then I probably (in general) don't want some other application to treat a name with just the first 32 bits of that as referring to the same thing, since the 32-bit name will have lots of colliding objects. If ni or nih URIs become widely used, there will be many cases where names will occur more than once in application protocols, and it'll be unpredictable which instance of the name would be used for name-data integrity checking, thus leading to threats. For this reason, we require that the algorithm, length, and value all match before we consider two names to be the same.

The fact that an ni URI includes a domain name in the authority field by itself implies nothing about the relationship between the owner of the domain name and any content referenced by that URI. While a name-data integrity service can be provided using ni URIs, that does not in any sense validate the authority part of the name. For example, there is nothing to stop anyone from creating an ni URI containing a hash of someone else's content. Application developers MUST NOT assume any relationship between the registrant of the domain name that is part of an ni URI and some matching content just because the ni URI matches that content.

If name-data integrity is successfully validated, and the hash is strong and long enough, then the "web origin" [RFC6454] for the bytes of the named object is really going to be the place from which you get the ni name and not the place from which you get the bytes of the object. This appears to offer a potential benefit if using ni names for scripts included from a HTML page accessed via server-authenticated https, for example. If name-data integrity is not validated (and it is optional) or fails, then the web origin is, as usual, the place from which the object bytes were received. Applications making use of ni names SHOULD take this into account in their trust models.

Some implementations might mishandle ni URIs that include non-base64 characters, whitespace, or other non-conforming strings, and that could lead to erroneously considering names to be the same when they are not. An ni URI that is malformed in such ways MUST NOT be treated as matching any other ni URI. Implementers need to check the behaviour of libraries for such parsing problems.

11. Acknowledgments

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 Carsten Bormann, Martin Duerst, Tobias Heer, Bjoern Hoehrmann, Tero Kivinen, Barry Leiba, Larry Masinter, David McGrew, Alexey Melnikov, Bob Moskowitz, Jonathan Rees, Eric Rescorla, Zach Shelby, and Martin Thomas for their comments and input to the document. Thanks, in particular, to James Manger for correcting the examples.

12. References

12.1. Normative References

[ISOIEC7812] ISO, "Identification cards -- Identification of issuers -- Part 1: Numbering system", ISO/IEC 7812-1:2006, October 2006, <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39698>.

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

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

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[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.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.

[SHA-256] NIST, "Secure Hash Standard", FIPS 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.

12.2. Informative References

[CCN] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09), December 2009.

[DECPARAMS] Hallam-Baker, P., Stradling, R., Farrell, S., Kutscher, D., and B. Ohlman, "The Named Information (ni) URI Scheme: Optional Features", Work in Progress, June 2012.

[Luhn] Wikipedia, "Luhn mod N algorithm", September 2011, <http://en.wikipedia.org/w/index.php?title=Luhn_mod_N_algorithm&oldid=449928878>.

[Magnet] Wikipedia, "Magnet URI scheme", March 2013, <http://en.wikipedia.org/w/index.php?title=Magnet_URI_scheme&oldid=546892719>.

[NETINF-ARCHITECTURE]
Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., and M. Karl, "Network of Information (NetInf) - An information-centric networking architecture", Computer Communications, Volume 36, Issue 7, pages 721-735, ISSN 0140-3664, 1 April 2013, <http://www.sciencedirect.com/science/article/pii/S0140366413000364>.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4773] Huston, G., "Administration of the IANA Special Purpose IPv6 Address Block", RFC 4773, December 2006.

[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.

[RFC5513] Farrel, A., "IANA Considerations for Three Letter Acronyms", RFC 5513, April 1 2009.

[RFC6149] Turner, S. and L. Chen, "MD2 to Historic Status", RFC 6149, March 2011.

[RFC6150] Turner, S. and L. Chen, "MD4 to Historic Status", RFC 6150, March 2011.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

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

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, November 2012.

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
EMail: kutscher@neclab.eu

Christian Dannewitz
University of Paderborn
Paderborn
Germany
EMail: cdannewitz@googlemail.com

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