Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: November 1, 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 30, 2012

Naming Things with Hashes
draft-farrell-decade-ni-05

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

Copyright Notice

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. This document specifies standard ways to do that to aid interoperability.

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 integrity of the name-data 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 [I-D.hallambaker-decade-ni-params] 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 segment 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 (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. 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 the decoded hashes are the same length and have the same binary value.

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.

Other than in the above 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 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 refer to the same object if the digest algorithm and value are the same.

One slash [Required] The literal "/"

Digest Algorithm [Required] The name of the digest algorithm, as specified in the IANA registry defined in Section 9.4 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 [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.

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

The query segment of a 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 (e.g., '%2f' or '%2F' for the slash character). Section 3.4 of [RFC3986] states that "The characters slash ("/") and question mark ("?") may represent data within the query component." All of this is as per RFC 3986, and should anything here conflict with that, RFC 3986 rules apply.

Note that when mapped to HTTP or HTTPS URLs, '/' and '?' characters in a query string will have to be percent encoded.

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 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 2: URI syntax

Adapting that for the Named Information URI:

```
         NI-URI   = ni-scheme ":" ni-hier-part [ "?" ni-query ]
                       ; adapted from "URI" in RFC 3986
         ni-scheme   = "ni"

         ni-hier-part   = "//" authority path-algval
                        / path-algval
                        ; adapted from "hier-part" in RFC 3986

         path-algval = "/" alg ";" val
         alg = 1*unreserved
         val = 1*unreserved

         ni-query    =  attr "=" value  [*( "&" attr "=" value )]
         attr           = query-token
         value        = query-token

         query-token = *( unreserved / pct-encoded )

         unreserved  = ALPHA / DIGIT / "-" / "." / "_" / "~"
             ;  directly from RFC 3986
             ; "authority" and "pct-encoded" are also from RFC 3986
```

Figure 3: ni Name syntax

Note that "unreserved" is defined in the URI specification [RFC3986] Section 2.3, in the way shown above. The "authority" and "query" types are also from 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 [RFC2616] or HTTPS [RFC2617] 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 retriever 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 Segment 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. [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 "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 whatever is being identified by the hash value. The format of the binary representation of a name is shown in Figure 4.

```
     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 4: 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 9.4 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) URI format is defined with scheme name "nih", standing for "Named Information for Humans." (Or possibly "Not Invented Here," which is clearly false, and therefore worth including :-)

As with the ni URI format, nih URI fields are separated by a semi-colon (;) character. The first field is a hash algorithm string, as in the ni URI format. The hash value is represented using lower-case

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 lower-case letters.

The hash value is OPTIONALLY followed by a checksum. The checksum MUST be calculated as a crc16 over the following parts (encoded as UTF-8 [RFC3629]): the URI scheme and separator ("nih:"), the algorithm string, the first delimiter, (";") the hash value, and the second delimiter (also ";"). The 16-bit result of the crc16 is encoded using network byte order and, like the hash value, with lower-case ASCII hex characters.

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

```
humanname = "nih:" algval [ ";" checksum ]
algval = alg ";" val
alg = 1*unreserved
val = 1*unreserved
checksum = 1*unreserved
```

Figure 5: Human-readable syntax

For algorithms that have a Suite ID reserved (see Figure 8), the alg field MAY contain the ID value as a UTF-8 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 presentation. Note that using the decimal presentation instead of the hash name string results in a different checksum for the same name.

8. Examples

The following ni URI references the text "Hello World!" (without the quotes, being 12 characters), 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
```

Given the SubjectPublicKeyInfo in Figure 6 we derive the names shown in Figure 7 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 6: 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-readable form of a name for this key (truncated to 120 bits |
| in length) with checksum:                                            |
| nih:sha-256-120;53269057e12fe2b74ba07c892560a2;8628                  |
+----------------------------------------------------------------------+
| Human-readable form of a name for this key (truncated to 32 bits  |
| in length) with checksum:                                            |
| nih:sha-256-32;53269057;5cab                                         |
+----------------------------------------------------------------------+
| Human-readable form using decimal presentation of the                |
| algorithm ID (sha-256-120) with checksum:                            |
| nih:3;53269057e12fe2b74ba07c892560a2;1535                            |
+----------------------------------------------------------------------+
```

Figure 7: Example Names

9. IANA Considerations

9.1. Assignment of Named 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 CoAP 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 Named Information for Humans (nih) 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: 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 with initial use cases provided by CoAP 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

9.3. 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

9.4. Hash Name Algorithm Registry

IANA is requested to create a new registry for hash algorithms as used in the name formats specified here. This registry has five fields, the binary suite ID, the hash algorithm name string, the truncation length, the underlying algorithm reference and a status field that indicates if algorithm is deprecated and should no longer be used. If the status is not "deprecated," then that does not necessarily mean that an algorithm is "good" for any particular purpose, since the cryptographic strength requirements will be set by other applications or protocols. The expert SHOULD seek IETF review before approving a request to mark an entry as "deprecated."

Future assignments are to be made through expert review [RFC5226]. Initial values are specified below. The expert SHOULD generally approve additions that reference hash algorithms that are widely used in other IETF protocols. In addition, the expert SHOULD NOT accept additions where the underlying hash function (with no truncation) is considered weak for collisions.

Since there are only 64 possible binary suite ID field values allowed by the binary format specified here, the suite ID field value is OPTIONAL. 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 expert may allow for later allocation of a suite ID, if that appears warranted. The expert MAY request IETF review before allocating a suite ID.

ID	Hash name string	Value length	Reference	Status
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 8: 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, is used for generating the hash. The document referenced for the hash algorithm MUST be such that it would be acceptable were the "specification required" rule applied.

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 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 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 will 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, Tobias Heer, Martin Thomas and James Manger 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.

[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.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[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.

[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.

12.2. Informative References

[I-D.hallambaker-decade-ni-params] Hallam-Baker, P., Stradling, R., Farrell, S., Kutscher, D., and B. Ohlman, "The Named Information (ni) URI Scheme: Optional Features", draft-hallambaker-decade-ni-params-02 (work in progress), April 2012.

[I-D.ietf-dane-protocol] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", draft-ietf-dane-protocol-20 (work in progress), April 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>.

[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