ENUM
Internet-Draft
Intended status: Experimental
Expires: January 13, 2009

B. Timms
J. Reid
Telnic
J. Schlyter
Kirei AB

July 12, 2008

# IANA Registration for Encrypted ENUM
# <draft-timms-encrypt-naptr-01.txt>

Status of this Memo

Abstract

This document requests IANA registration of the "X-Crypto" Enumservice. This Enumservice indicates that its NAPTR holds a Uniform Resource Identifier that carries encrypted content from the fields of another (unpublished) Protected NAPTR, for use in E.164 Number Mapping (ENUM).

Table of Contents

# 1. Introduction

## 1.1. The problem

The Domain Name System or DNS ([RFC1034],[RFC1035]) is a global system; it does not differentiate on the data it returns. If a Naming Authority Pointer (NAPTR) resource record [RFC3403] is published in DNS, then by definition the same Resource Record Set (RRset) will be returned in response to a query, regardless of the user placing that query.

Where Universal Resource Indicators (URIs, defined in [RFC3986]) are published within DNS (inside NAPTRs), the registrant may prefer to make these available only to groups of individuals that he or she has selected. Given the global nature of DNS, this can be a problem.

It is not reliably possible to return different RRset content to different queries, depending on the user making the request. Even if the authoritative server has been configured to discriminate based on the source of the query, if there are any intermediary recursive resolvers, the query may not even be passed to the authoritative server and the response returned to the querying DNS client may not be as the authoritative server would have chosen. It can also be challenging to configure and maintain the authoritative server, and this may also involve special configuration of each client that will query for and use the data.

### 1.1.1. The requirements

There should be no need to use any special configuration for the authoritative name servers, clients, or any intermediary recursive resolvers when using this scheme. Also, there should be no special DNS processing for the resource records used in any DNS components. As a secondary requirement that follows from these, the same content should be published in DNS and so made available to all without discrimination. This will match the distributed design of the DNS and maintain the effectiveness of the cacheing architecture.

However, the value of chosen content should be protected in such a way that it is understandable only by a selected set of users.

There should be no performance impact on those clients that choose not to process protected data. Thus it is important that the recipient of this data can detect immediately that it is protected, and either process it to extract the protected content using its private knowledge, or immediately discard the data if it is not interested in such protected records.

### 1.2. The solution

A general solution for all DNS resource records that meets these requirements is very difficult; the performance requirements for DNS in general are severe. NAPTRs stored in ENUM [RFC3761] domains may contain personally identifying information, so finding a solution may be considered more pressing for such NAPTRs, and some restrictions or processing costs may therefore be acceptable. Also, in the case of NAPTRs a solution may be possible, as the problem is more restricted. NAPTRs hold a small number of well defined fields. Not all of these fields in a NAPTR will be sensitive and so require protection.

Those fields to be protected can be encrypted using a key known to the intended users. Thus a "Protected" NAPTR can be processed into two parts; the protected fields carried in a ciphertext, and the public fields. A "Container" NAPTR can itself be used to carry this ciphertext (inside its Regexp field content, in a URI), along with those fields that are considered public and are not protected. These public fields can be copied from the Protected NAPTR into this Container NAPTR.

The Container NAPTR can be stored and retrieved in the normal way. It will have an Enumservice indicating that it acts as a container and MUST be decoded before the original Protected NAPTR can be reconstructed and processed. When an ENUM client retrieves such a Container NAPTR, it can immediately know that this requires cryptographic processing, and if that client is not interested in such processing, the NAPTR can be discarded.

#### 1.2.1. Protected fields

There is no great benefit to encrypting all of the RDATA (Resource record Data) for a NAPTR. The ORDER and PREFERENCE/PRIORITY fields are used to indicate the preferred order in which the records within a returned NAPTR RRset will be processed. Whether a particular NAPTR acts as a container for a Protected NAPTR's content or not, the order in which it will be processed should remain the same; there is no change to the Dynamic Delegation Discovery System (DDDS) algorithm specified in [RFC3402].

If instead the Container NAPTR had a different ORDER and PREFERENCE/PRIORITY field values from those held in the Protected NAPTR, it might be possible for a Container NAPTR to be considered first (as it had a low numerical order/preference value), but, once decoded, the Protected NAPTR content it contained was of a much lower priority, and so should not be processed at that point. This would be inappropriate, and so the ORDER and PREFERENCE/PRIORITY fields will remain the same in both Protected and Container NAPTRs.

Thus the ORDER and PREFERENCE/PRIORITY field values should be considered public and be copied from the Protected NAPTR into the Container NAPTR. However, all the other fields (flags, services, regexp, and replacement) are sensitive in nature, and so that portion of the binary format of the RDATA in which they are held will form the plaintext that will be protected before publication in DNS.

### 1.2.2. Protection process

The NAPTR to be protected can have these sensitive fields placed into a plaintext buffer. The buffer content is then encrypted using an appropriate key to create a ciphertext. The ciphertext can then be "armoured" into a form that can be presented in a data URI [RFC2397]. That URI can then be placed in a Container NAPTR within its Regexp field, along with the "public" ORDER and PREFERENCE/PRIORITY fields copied from the Protected NAPTR, together with a dedicated Enumservice, terminal URI flag field ('u') and an empty Replacement field.

If this Container NAPTR has an appropriate Enumservice then its nature will be immediately detectable by the recipient of that NAPTR. If the recipient has the appropriate key to decode the URI data, then it can decrypt the URI content to form a buffer with the plaintext fields. These fields (in combination with the ORDER and PREFERENCE/PRIORITY fields that have been copied into the Container NAPTR and have not been encoded) can be reconstructed into the RDATA that would have existed in the original Protected NAPTR. That Protected NAPTR content can then be processed by the client in the normal way.

## 2. Terminology

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 [RFC2119].

## 3. Enumservice Registration - X-Crypto

The following template contains information required for the IANA registrations of the 'X-Crypto' Enumservice, according to Section 3 of RFC 3761:

Enumservice Name: "X-Crypto"

Enumservice Type: "x-crypto"

Enumservice Subtype: "data"

Enumservice Sub-subtype: see Section 5

URI Schemes: "data"

Functional Specification: see Section 4

Security Specification: see Section 7

Intended Usage: COMMON

Author(s): Ben Timms, Jim Reid, Jakob Schlyter. (for authors contact details see Authors' Addresses section).

Any other information that the author deems interesting: None

## 4. Functional Specification

The basic concept is covered in Section 1.2 and the process is covered in Section 1.2.2. This section describes in detail how each of the fields are handled.

Publication and use of a NAPTR with this Enumservice is based on two concepts:

A Protected NAPTR that contains sensitive field values, and is not stored and published in DNS.

A Container NAPTR with this Enumservice that holds the protected fields in encrypted form within its Regexp field. This NAPTR carries the "x-crypto" Enumservice

The Container NAPTR resource record fields are as follows:

### 4.1. Order

The value of the order field is copied in clear from the RDATA of the Protected NAPTR into the Container NAPTR. It is not encoded.

### 4.2. Preference

The value of the preference field is copied in clear from the RDATA of the Protected NAPTR to the Container NAPTR. It is not encoded.

### 4.3. Services

The value of the services field for the Container NAPTR is set to "E2U+x-crypto:data:" combined with the ciphersuite sub-subtype (Section 5).

### 4.4. Regexp

The encrypted payload (Section 4.6) is encoded in Base64 [RFC4648], and transported as the value of a data URI [RFC2397] inside the Container NAPTR.

Container NAPTR Regexp Example:
!^.*$!data:;base64,bWVrbWl0YXNkaWdvYXQK!

### 4.5. Replacement

The value of the Container NAPTR's replacement field MUST be set to ".".

### 4.6. Encrypted Payload Generation

The Encrypted Payload consists of a base64 "armoured" string holding an encrypted, optionally padded ciphertext reflecting a portion of the Protected NAPTR's RDATA.

The portion of the Protected NAPTR RDATA holding its Flags, Services, Regexp and Replacement fields is treated as the plaintext to be processed. This plaintext is (optionally) padded and the resulting block is then encrypted, to form the ciphertext. Potentially, an optional hash may be generated from this. Once this cipehertext has been generated, it is further encoded in Base64 to form the encrypted payload that is then used as the value of the Container NAPTR's URI.

## 5. Ciphersuite Subtypes

The enumservice sub-subtype carries the ciphersuite used for the encrypted payload.

Ciphersuite sub-subtype example: RSA 1024-bit with PKCS#1.5 padding and no hash would be encoded as 0x8210 and presented as enumservice "E2U+x-crypto:data:8210".

### 5.1. Crypto Algorithms

A 1-byte field indicating the encryption algorithm is used for the encrypted payload (Section 4.6).

| Value | Encryption Algorithm |
|---|---|
| 0x00 | NULL |
| 0x81 | RSA-512 |
| 0x82 | RSA-1024 |
| 0x83 | RSA-1536 |
| 0x84 | RSA-2048 |
| 0x85 | RSA-3072 |
| 0x86 | RSA-4096 |

### 5.2. Padding

A 4-bit field indicating what padding algorithm is used for the encrypted payload (Section 4.6).

| Value | Padding Algorithm |
|---|---|
| 0x0 | NULL |
| 0x1 | PKCS #1.5 |
| 0x2 | OAEP |

### 5.3. Hash

A 4-bit field indicating what hash algorithm is used for the encrypted payload (Section 4.6).

| Value | Hash Algorithm |
|---|---|
| 0x0 | NULL |
| 0x1 | MD2 |
| 0x2 | MD5 |
| 0x3 | SHA-1 |
| 0x4 | SHA-224 |
| 0x5 | SHA-256 |
| 0x6 | SHA-384 |
| 0x7 | SHA-512 |

## 6. Examples

In these examples, a 1024-bit RSA key pair is used for encryption and decryption. The plaintext has cryptographic padding applied prior to encryption, using the PKCS 1.5 algorithm. There is a null hash applied to this. The ciphersuite value (i.e. the Enumservice sub-sub-type string held in the container NAPTR) will be '8210'. The resultant ciphertext is further "armoured" using Base 64 encoding, and is placed into a data URI in that container NAPTR. A "default" or "greedy" Extended Regular Expression, or ERE (i.e. '!^.*$!') is used in the container NAPTR Regexp field. Note that (in keeping with [RFC4648], section 3.1), the "aromouring" process MUST NOT add line feeds to the base-encoded data.

For ease of presentation the examples are shown in textual form, but the encryption process works on the binary form of the RDATA fields.

Note that, using this encryption, encoding and ERE mechanism, the limit to the length of the "plaintext" (i.e. the fields of the RDATA to be protected) is 117 bytes. This is because RSA is a block cipher, with in this case 1024 bits (128 bytes) as the block size, and the PKCS #1.5 padding to be added to the plaintext consumes 11 bytes, leaving a maximum plaintext size of 117 bytes.

Once encrypted, the ciphertext is also 128 bytes, but this is in binary form, and has to be further "armoured" for carriage inside the data: URI. The Base 64 encoding process expands the ciphertext to 4/3 (rounded up to the nearest 4 bytes) of its size in binary form, giving a text block of 172 bytes. To this has to be added the rest of the container NAPTR's Regexp content, giving a Regexp field length of 193 bytes. Note that this field length is constant, as the ciphertext is always the same length when using the same encryption and padding scheme (regardless of the length of the plaintext fields to be protected and whether these fields reflect a terminal or non-terminal NAPTR).

In the following examples, it is assumed that the private key (in PKCS 8 format, protected with the pass phrase "pkcs8passphrase", and expressed in PEM format), is:

```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIICoTAbBgkqhkiG9w0BBQMwDgQIpwb76GrK0AgCAggABIICgEsRtkQ2isuKq3Cl
8wpAfDxzzFbumj0HdGu7WLEElVYaLt4CvRlz5kL3SK2G8ydpsdU104s0RnzgPGFv
63zsJZ5bC6d4lCcWMjbhv+U91YUhlrc6R6UHhIN4BSBWTeA/Ia/U7bwZm/TV7ke1
eeLOdsyzEnPr9lj3v1wdHFYU6CMY1lRP/lbqexVQMYEev5/w+tu9LyGdP3MPnnUC
N/OVk67OkwBqrBnQpHtHLC7i0bq6srLaJWB7nFVa/iXlQ8lCwFRGwV7RDq6JvgI2
LjgJArLG8i7QBx+WE/+LwLAessmU4RUzzvQhnlWdf3UkXR/hRDPRhJFFrr5zjfUr
cF5CfZms4WhMh+epqnlaoaX7uEj1gQ9gLvNef3b68akpRVIFuLBnqcdbahr0rzQA
```

```
/LT5e62jfo2saSG9vVAY7f1fOmrh9K9+S69rlbQ4JLzx+sttPTty9CscIZf3/cDM
1GNdumwG1HewzkVuzvIlyJfLbM1ftWhv0tWTe8pPFqccTFYvwpjVnOxN6yr8EMUy
8VTyDCZT/us8E6vtei2fbvLHnmRzIqCgHUAYBVKM+cwrGELCuSBbx6pmOp21EyGH
5+Lobg0XYjArVxgynNvAU/mmQWbdVqhkfrIGhywCvi1+Jpigvn+2zBayCfPitdxh
BjvYhEp6qSE0/QWog4Qn6t5TaDK7ddV39Tw2pyuE1Gh+tAfZWrwO0aF9NKI9G5mJ
Db3Jqm9UzAoulH8cnMdCAa7oDVCl/8ky1VKTIz8fe3bVsMCm8Cgv3/vz8vupaGV5
exzJUEeEJweenReOaI8Eocl2qSKmcrtlhAQI+l77KnvM3J0QSPSxeH203OnLovG3
lQkJjd4=
-----END ENCRYPTED PRIVATE KEY-----
```

The public key, expressed in PEM format, is:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDoOyVQpfJlai3i7RlF5iGGlYUb
/HXOuyV1IVKoQ1iQQvm91gU5L10GwVWN1WY4yfqYtPXnJMoMbAIK72wNnaB6Jo4/
ELbi40yOQSIe4TKXVcfMkFbpJlN7FfktHtpLai60zsT8Ywt4OF8rUFmb5CdE3gtV
yqQfmfczYheXqPW7iwIDAQAB
-----END PUBLIC KEY-----
```

### 6.1. Terminal NAPTR Example

This example shows a NAPTR holding a SIP Enumservice as the protected NAPTR. Given that a terminal E2U (ENUM) NAPTR is to be protected, the combined maximum length of the Enumservice string, the ERE pattern and the original URI is (117 - 12) bytes, or 105 bytes. If a "greedy" ERE field is used in that protected NAPTR, the space available for the Enumservice string and the URI is 105 - 4 bytes, or 101 bytes. In the example shown here, the NAPTR to be protected has a SIP Enumservice (and corresponding sip: URI scheme). This means that the SIP address value can be a maximum of 94 bytes long. In this case, it occuplies 24 bytes.

The protected NAPTR RDATA:

```
100 50 'u' 'E2U+sip' '!^.*$!sip:alice@wonderland.example!' .
```

Is replaced by the container NAPTR RDATA:

```
100 50 'u' 'E2U+X-crypto:data:8210'
'!^.*$!data:;base64,OQZl3x9TEaZtQamA5t3IJqXaKUT6QuV+yLtW34/hszd
D2jtSwavlxiax8CDMWekikXkgbPQqEo7X6g8aX3REiXVJ/PqrbxFASIIktnIIVE
rZU3RVl8WAvxQvWGs+wEY3YAi4UnOoqAOdbv3tsV0i4h15I+wePz9Rw9VBpU95h
Wc=!' .
```

### 6.2. Non-Terminal NAPTR Example

In this example, a non-terminal NAPTR is protected. As the replacement field in a NAPTR is not permitted (as specified in RFC 3403) to use DNS domain compression, the fully qualified domain name of the target domain is held in the replacement field. This fully qualified domain name is, of course, stored in binary form within the RDATA.

Note that, using the same protection scheme (1024-bit RSA, with PKCS 1.5 padding, null hash), the maximum length of the fully qualified domain name will be 117 - 3, or 114 bytes. In this example, the fully qualified domain name is 1+5+1+10+1+7+1 = 26 bytes long.

The protected non-terminal NAPTR RDATA:

```
100 51 '' '' '' alice.wonderland.example.
```

Is replaced by the container NAPTR RDATA:

```
100 51 'u' 'e2u+x-crypto:data:8210'
'!^.*$!data:;base64,j+WNPPwriy5pu4SfabMavRtE+c/f3Sk62Ab5TNYOomo
RcGrKk5q23i6BB4fp71+z3ezK1U91jTdpzmpF0M7WVs9M9AnhDxyrbQwo1mP/uU
YpflaZuG5aEnY14aTntAldh7UacPvfaiWc1QPg/6C9Wb7MBedmZAYajc2YZHgKQ
1o=!' .
```

## 7. Security Considerations

This is an Enumservice for a NAPTR intended to carry protected NAPTR content in encrypted form. It does not discuss the means by which the keys needed to decrypt the protected content are exchanged. For this Enumservice registration, this is considered "out of scope". However, the technique used for key exchange is important, and must be considered thoroughly; there is little point in applying a complex encryption scheme if the keys are available to eavesdroppers. Of course, although technically permitted, confidentiality will not be achieved unless a non-null encryption is applied.

There are limitations on field size within DNS, so that, for example, the Regexp field has a maximum length of 255 bytes. Several of these bytes will be taken up in the container NAPTR's Regexp field with the sub-field delimiters, with the ERE sub-field content, and with the URI scheme itself. There is limited space to carry the armoured ciphertext as the data URI value, given the armouring choice proposed here and the simple use of the existing Regexp field to carry the protected data. This will in turn limit the choices for encryption method, hash algorithm, and any padding. See also the examples above.

Even if an eavesdropper cannot decode the content carried in a container NAPTR, the fact that ORDER and PREFERENCE/PRIORITY fields are copied from the protected NAPTR opens the possibility of opaque traffic analysis. If the ORDER and PREFERENCE/PRIORITY field values for a NAPTR within a RRSet change (for example, to reflect the domain owner going from office to home) then this change will be reflected in the NAPTR even if it is protected. Simply due to changes in RRSet relative ORDER and PREFERENCE/PRIORITY values, an attacker might surmise that the protected data was associated with the domain owner's office or home number. This information might in itself be useful to the attacker (for example, by indicating when the domain owner was or was not present at a particular location).

Finally, if the value of a contact (such as a SIP URI or telephone number) were to be available to an attacker using other means, then there may be potential for differential cryptographic analysis based on an assumed "known plaintext". The amount of data available to an attacker with realistic numbers of NAPTRs is small, but it is important to use appropriate cryptographic padding to limit the potential for such an attack.

These issues mean that the environment in which NAPTRs with this Enumservice can be used may be restricted, and further security analysis will depend on deployment experience.

An analysis of threats specific to the dependence of ENUM on the DNS, and the applicability of DNSSEC ("Domain Name Security") [RFC4035] to these, is provided in [RFC3833].

## 8. IANA Considerations

This document requests registration of the "X-Crypto" Enumservice with the "x-crypto:data:<ciphersuite>" type according to the guidelines and specifications in RFC 3761 [RFC3761] and the definitions in this document. This Enumservice is intended for use with the "data:" URI scheme.

## 9. Acknowledgements

The authors gratefully acknowledge the contributions of Romek Szczesniak. He helped greatly to clarify some of the issues with deployed security schemes and current implementations. We also acknowledge the support of Khashayar Mahdavi whose original idea this draft embodies, and Henri Asseily for driving the development of the environment in which this is being used.

## 10. References

### 10.1. Normative References

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

[RFC2397] Masinter, L., "The "data" URL scheme", RFC 2397, August 1998.

[RFC3402] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm", RFC 3402, October 2002.

[RFC3403] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

[RFC3761] Faltstrom, P. and M. Mealling, "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)", RFC 3761, April 2004.

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

### 10.2. Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC3833] Atkins, D. and R. Austein, "Threat Analysis of the Domain Name System (DNS)", RFC 3833, August 2004.

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

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

Authors' Addresses

Ben Timms
Telnic
37 Percy Street
London W1T 2DJ
United Kingdom

Email: btimms@telnic.org

Jim Reid
Telnic
37 Percy Street
London W1T 2DJ
United Kingdom

Email: jim@telnic.org

Jakob Schlyter
Kirei AB
P.O. Box 53204
Goteborg SE-400 16
Sweden

Email: jakob@kirei.se

## Full Copyright Statement



## Intellectual Property