Abstract

The DNS is the last common Internet protocol that has no encryption scheme and therefore provides no privacy to the users. This document proposes an extensible mechanism providing encryption of DNS queries and responses with method for secure retrieval and verification of validity of encryption keys. It is independent of the underlying transport protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

Krecicki                 Expires April 21, 2016
Internet-Draft                          Internet Systems Consortium
Intended status: Standards Track          October 19, 2015
Expires: April 21, 2016
Stateless DNS Encryption                  
draft-krecicki-dprive-dnsenc-01
The Domain Name System protocol is specified in [RFC1034] and [RFC1035]. DNS messages are unencrypted and therefore prone to eavesdropping. Although it’s considered only metadata, there is a lot of information that can be leaked, as explained by [RFC7626].

The DNS protocol is very lightweight - the queries are usually less than 100 bytes long and the responses are usually less than 1000 bytes (even with DNSSEC). Existing transport encryption schemes such as TLS for TCP or DTLS for UDP give a huge and unnecessary overhead both in the amount of data sent and retrieved and in the number of packets exchanged between client and server.
In DNSENC the query is encrypted using asymmetric cryptography with a securely retrieved key and the response is encrypted using symmetric encryption with a one-time key provided with the query. The DNSENC protocol is uses standard DNS features and does not requires any additional external mechanisms such as a PKI/CA system.

The DNSENC communication can be split into three phases:

- first the client retrieves public key for the server that is stored in DNS and is DNSSEC signed; this key can then be cached,
- the client creates the query, adds a random response encryption key and encrypts the query with the server’s public key,
- the server decrypts the message, prepares the response and encrypts it with the key provided by client.

It has to be noted that for a resolver to bootstrap itself, it has to perform a clear text query to retrieve keys for DNSENC encryption. As this query can be for information from the root zone, no sensitive information is leaked.

In an usual case DNSENC will add no additional round-trips to communication, besides the query for root zone servers’ NSK record. The message size overhead is around 140 bytes for encapsulation (NOTE: that’s using method described in Section 2.2.1), the maximum amount of overhead for encryption depends on the method used and varies from 16 bytes for AES to 512 bytes for RSA.

1.1. Requirements Language

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

2. Communication process

2.1. Key retrieval

To communicate securely with a server, a client first needs to retrieve the server’s public key for asymmetric encryption. This key is stored in an NSK RR described in Section 3.1. The NSK RRset might contain multiple records. The key retrieval process is different for authoritative servers and for recursive servers.

For authoritative servers a NSK RRset SHOULD be present at a delegation point in the parent zone. If a NSK RRset is present at the delegation point, the name server MUST return both the NSK RRset
and its associated RRSIG RR(s) in the Authority section along with the NS RRset. This reduces the number of round-trips and allows all communication with the child zone’s servers to be encrypted.

All NSK RRsets MUST be signed with DNSSEC, and NSK RRsets MUST NOT appear at a zone’s apex.

example.com. 86400 IN NS a.iana-servers.net.
example.com. 86400 IN NS b.iana-servers.net.
example.com. 86400 IN NSK (...)
example.com. 86400 IN NSK (...)
example.com. 86400 IN RRSIG NSK (...)

For recursive servers and in cases where it is not possible to put the NSK RRset for an authoritative server at a delegation point, the NSK RRset SHOULD be present in the reverse DNS record of servers’ IP address, as described in [RFC3152], [RFC1033] and [RFC2317]. The client MUST retrieve this RRset from the server it wants to query. This RRset MUST be DNSSEC signed. For authoritative servers this method is only a fallback and client MUST try to retrieve the key at a delegation point first.

5.2.0.192.in-addr.arpa. 86400 IN NSK (...)

If the record is not signed the client MUST NOT use the key provided. (TODO enforce encryption, protection against downgrade attack).

As there might be multiple NSK RRs in the RRSet it is the client responsibility to choose the highest priority RR that has both query and response schemes supported by the client.

2.2. Query creation

-----------------------------------------------

NOTE: two alternatives are currently under discussion. Both are fully compatible with the DNS protocol. The first one is kind of a kludge but has better chances to be compatible with not-fully-protocol-compliant forwarders (although it still requires support for EDNS). The second solution is cleaner but might not work with some forwarders as the packet is not a regular DNS QUERY. The author prefers the first method.

-----------------------------------------------

After choosing an encryption scheme, the client generates a random response encryption key (symmetrical, eg. AES) and prepares a
regular DNS query with an NSK record containing the response encryption scheme and key in the ADDITIONAL section, eg:

```
+-----------------------------------------+
| Header          | OPCODE=QUERY, ID=997, QR=QUERY |
+-----------------------------------------+
| Question        | QTYPE=A, QCLASS=IN, QNAME=EXAMPLE.COM |
+-----------------------------------------+
| Answer          | <empty> |
+-----------------------------------------+
| Authority       | <empty> |
+-----------------------------------------+
| Additional      | . NSK IN NSK-RR |
+-----------------------------------------+
```

It has to be noted that this packet is never sent on the wire in raw form, so records in the ADDITIONAL section have no impact on compatibility with non-conforming forwarders.

2.2.1. Method 1

This message is encrypted using chosen query encryption key and packed, along with encryption key ID, in a DNSENC OPTION, as described in Section 3.2. A new query is created with:

- the query id copied from the encrypted message
- a <nonce>.dnsenc.arpa. TXT query in QUERY section, where <nonce> is a random label that guarantees that should the response be cached by a forwarder, it will not be reused for any other client.
- empty ANSWER and AUTHORITY sections
- a DNSENC OPTION in ADDITIONAL section

eg.:
...and sent to server. The response encryption key is stored on client along its identifier for decryption.

2.2.2. Method 2

This message is encrypted using query encryption key and packed inside a query with the OPCODE set to ENCRYPT:

```
  1  1  1  1  1  1
 0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|  ENCRYPT  |        Z           |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|               ENCRYPTED PAYLOAD LENGTH          |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|               ENCRYPTION KEY ID               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    RESERVED                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|               ENCRYPTED PAYLOAD               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

...and sent to server. The response encryption key is stored on client along its identifier for decryption.
2.3. Query processing

When a server supporting DNSENC receives a query for any name in the "dnsenc.arpa." domain it MUST immediately look in ADDITIONAL section for DNSENC OPTION. If the DNSENC OPTION is missing the server MUST respond with a status of REFUSED. The DNSENC OPTION is then decoded. If the server does not have a key with identifier provided in query or the decryption failed the server MUST respond with BADCRYPT response. If the data contained in the decrypted packet is invalid or the query ID is not the same as the encapsulating packet query ID the server MUST respond with FORMERR response.

A server that does not support DNSENC should respond with REFUSED response code, although old implementations not supporting [RFC6891] might return FORMERR.

A forwarder/proxy MUST pass the message untouched (along with the DNSENC OPTION in the ADDITIONAL section) - as described in [RFC5625]. A forwarder/proxy MUST NOT cache a result for anything within "dnsenc.arpa." domain.

2.4. Response creation

NOTE: this section describes creating the response for a query described in Section 2.2.1, the method for creating a response for a query described in Section 2.2.2 is analogous but will be described here iff WG decides to go ahead with this method.

After decryption, the NSK record from the query is saved and the query is processed normally. When the response is ready, a regular DNS response packet is created. This packet is encrypted using previously saved response encryption key sent by client and stored along with the response encryption key ID (to keep the response in the same format as query) in a DNSENC OPTION. A new response packet is created with:

- the query ID same as in encrypted packet
- the QUESTION section copied from original query
- an empty ANSWER and AUTHORITY sections
- a DNSENC Option in EDNS(0) OPT RR in the ADDITIONAL section containing the encrypted response to the "real" query.
This is then sent back to the client.

2.5. Response processing

TODO

3. Data types

3.1. NSK record

The NSK RR consist of priority field, key identifier, server names for which the key can be used (wildcard ‘*’ is supported, empty value means all servers for the zone), query encryption scheme (asymmetrical, eg. rsaes-oaep-2048), query key data and possible response encryption schemes. If the server provides multiple NSK records, it is the client’s responsibility to choose the highest priority NSK that has query and response encryption schemes supported by client.

3.1.1. Wire format

The NSK RR has following format:
where:

KEY IDENTIFIER  32-bit key identifier, unique for server

PRIORITy  8-bit key priority as preferred by server

APPLICABLE NS NAME  a <domain-name> of NSes that use this key. This allows for different NSes for a zone to use different keysets (eg. when the secondary is operated by different entity than primary). This field might contain wildcard symbol ‘*’ at any place (including as a part of a single label - eg. ‘ns*.foo*bar.example.com’), which matches zero or more characters and can cross label boundaries (‘ns*.example.com’).
matches ‘ns.example.com’,
‘ns1.example.com’ and
‘ns1.foobar.example.com’), single ‘*’
means any.

**ENCRYPTION SCHEME**
16-bit identifier of the encryption
scheme, as defined in [TBD IANA REGISTRY]

**KEY DATA**
encryption key - raw binary data

**RESPONSE ENCRYPTION SCHEMES**
list of 16-bit identifiers of response
encryption schemes, as defined in [TBD
IANA REGISTRY], in order of server
preference from most preferred

For NSK records retrieved by a resolver as described in Section 2.1
ENCRYPTION SCHEME describes the scheme used by client to encrypt the
query, which MUST be an asymmetric one. RESPONSE ENCRYPTION SCHEMES
MUST contain at least one scheme, and all encryption schemes MUST be
symmetric.

For NSK records sent in the ADDITIONAL section of a query and used by
the server to encrypt response:

- PRIORITY MUST be set to 0
- APPLICABLE NS NAME MUST be empty
- ENCRYPTION SCHEME MUST be symmetric and be one from the list of
  RESPONSE ENCRYPTION SCHEMES in a NSK record with key used to
  encrypt this query.
- KEY DATA MUST be a one-time, random key
- RESPONSE ENCRYPTION SCHEMES MUST be empty

3.1.2. Presentation format

All numerical fields are presented in decimal form, query key data is
base64 encoded, encryption scheme can be represented as a number or
as a name, fields are in the same order as in wire format, eg:
example.com. 86400 IN NSK (896417428 ; identifier
10 ; priority
ns*.example.com ; applicable NS name
1 ; query encryption
; scheme
"bnJfZnJlZV9wYWdlcyaMTU2NjgKbnJf
YWxsb2NfYmF0Y2ggNDE1OQpucl9pbmFj
dGl2ZV9hbm9uIDM1NzczNwpuc19hY3Rp
IDQ4OTAwMQpucl9kaXJ0eSxtMTUKbnJf
ZCAwCg==" ; query key, b64 encoded
32769 aes-256 ; response encryption
; schemes
}

3.2. DNSENC option

The encrypted packet is transmitted as an EDNS(0) option, with
OPTION-CODE set to DNSENC (value of [TBD]). The wire format of
OPTION-DATA for this option is:

+----------------------------------------+
| KEY IDENTIFIER                           |
+----------------------------------------+
| +----------------------------------------+ |
| | ENCRYPTED PAYLOAD |                     |
+----------------------------------------+    

The length of the encrypted payload is determined by the OPTION-
LENGTH field.

4. Encryption schemes

This document defines basic encryption schemes to be used for query
and response encryption. Those schemes SHOULD be implemented by all
servers and clients.

4.1. Asymmetric schemes

TBD: RSAES-1024-EOAP, ECC as a default?
4.2. Symmetric schemes

TBD: AES-256

5. Security Considerations

The security of this protocol is based deeply on DNSSEC [RFC4033]. Protection against downgrade attack requires wide adoption of DNSSEC.

6. IANA Considerations

NOTE: as for now no actions with IANA has been taken.

6.1. NSK RR type

This document defines a new DNS Resource Record Type, named "NSK". The IANA has assigned a code point from the "Resource Record (RR) TYPEs" sub-registry of the "Domain Name System (DNS) Parameters" registry for this record.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSK</td>
<td>TBD</td>
<td>Name Server Key</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

6.2. DNSENC EDNS option

This document defines a new EDNS option, named "DNSENC". The IANA has assigned a code point from the "DNS EDNS0 Option Codes (OPT)" sub-registry of the "Domain Name System (DNS) Parameters" registry for this option. NOTE: this is required only for Section 2.2.1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Status</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>[TBD]</td>
<td>DNSENC</td>
<td>Standard</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

6.3. ENCRYPT OpCode

This document defines a new DNS OPCODE, named "ENCRYPT". The IANA has assigned a code point from the "DNS OpCodes" sub-registry of the "Domain Name System (DNS) Parameters" registry for this opcode. NOTE: this is required only for Section 2.2.2.

<table>
<thead>
<tr>
<th>OpCode</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>ENCRYPT</td>
<td>[this document]</td>
</tr>
</tbody>
</table>
6.4. BADCRYPT RCODE

This document defines a new DNS RCODE, named "BADCRYPT". The IANA has assigned a code point from the "DNS RCODEs" sub-registry of the "Domain Name System (DNS) Parameters" registry for this opcode.

<table>
<thead>
<tr>
<th>OpCode</th>
<th>Name</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>BADCRYPT</td>
<td>Encryption error</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

6.5. DNSENC encryption schemes registry

The IANA has created and maintains a sub-registry (the "DNSENC encryption schemes" registry) of the "Domain Name System (DNS) Parameters" registry. The initial values for this registry are below.

An "Expert Review" [RFC5226] is required for the assignment of new scheme value (TBD: Expert or IETF? We should have a proper definition of an encryption scheme).

This registry holds a set of 16-bit values identifying an encryption scheme. First bit of first octet is set to 0 for asymmetric encryption schemes, 1 for symmetric encryption schemes. First nibble of second octet is set to 0xf for experimental or vendor-specific schemes.

The initial assignments in this registry are:

<table>
<thead>
<tr>
<th>Octet 1</th>
<th>Octet 2</th>
<th>Algorithm</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>Reserved</td>
<td>[this document]</td>
</tr>
<tr>
<td>0x00</td>
<td>0x01</td>
<td>RSAES-2048-OAEP</td>
<td>[this document]</td>
</tr>
<tr>
<td>0x00</td>
<td>0x02</td>
<td>RSAES-4096-OAEP</td>
<td>[this document]</td>
</tr>
<tr>
<td>0x80</td>
<td>0x00</td>
<td>Reserved</td>
<td>[this document]</td>
</tr>
<tr>
<td>0x80</td>
<td>0x01</td>
<td>AES-256</td>
<td>[this document]</td>
</tr>
<tr>
<td>0x80</td>
<td>0x01</td>
<td>AES-512</td>
<td>[this document]</td>
</tr>
<tr>
<td>....</td>
<td>0xf.</td>
<td>experimental</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>or vendor-specific</td>
<td></td>
</tr>
</tbody>
</table>

7. Normative References


Author’s Address

Witold Krecicki
Internet Systems Consortium
950 Charter Street
Redwood City, CA  94063
USA

Email: wpk@isc.org
URI:  http://www.isc.org