Network Working Group	A. Langley
Internet-Draft	Google Inc
Expires: January 02, 2012	July 01, 2011

Serializing DNS Records with DNSSEC Authentication
draft-agl-dane-serializechain-01

Abstract

This document describes a format for serializing a DNS record with accompanying DNSSEC information such that a verifier can be convinced that the DNS record is authentic without performing DNS queries itself.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

RFC 1035 *[RFC1035]* describes the Domain Name System, a distributed database. For many years the authenticity of the contents of this database were founded on trust and small amounts of entropy in queries that frustrated efforts of attackers to blindly generate acceptable responses. In RFC 4033 *[RFC4033]* (and many others), a system of cryptographic signatures for DNS (called DNSSEC) was specified. DNS clients can now query DNS servers and strongly establish the authenticity of the resulting responses.
This memo specifies a format for serializing a DNS record and all the DNSSEC signatures needed to strongly establish the authenticity of that record. This may be useful for clients who cannot query the DNS directly, or for when such queries would not be performant.

2. Requirements Notation

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]*.

3. Overview

The serialization holds few surprises and most of the data structures share a great deal with the wire format of the DNS records from RFC 4034 *[RFC4034]*. The only tricks are size optimizations: the root key and the content of some DS records can be omitted. Additionally, for zones which don't use a ZSK and KSK, the signature over the DNSKEY can be omitted.
All integers described are unsigned and big-endian. byte is used an a synonym for an unsigned, 8-bit integer when such data might be better

thought of as opaque octets. There is no alignment; all structures are packed.

```
struct Signature {
  uint16 length;
  uint8 algorithm;
  uint8 labels;
  uint32 ttl;
  uint32 expires;
  uint32 begins;
  uint16 key_tag;
  byte signature[length - 16];
};
```

A Signature structure is the RRSIG RDATA wire format from RFC 4034 *[RFC4034]* section 3.1 with the Type Covered and Signer's Name omitted. These values are implied by the context when used in this serialization format and aren't duplicated for space reasons.
The length member may be zero to signify that a signature has been omitted. In this case, no further members of the structure will follow.

```
struct Key {
  uint16 length;
  byte rdata[length];
}

struct RRData {
  uint16 length;
  byte rrdata[length];
}
```

Resource records are serialized using exactly the DNS wire format. Keys, likewise, are exactly as specified in RFC 4034 *[RFC4034]* section 2.1.

```
struct Ds {
  uint8 digest_type;
  uint16 digest_len;
  byte digest[digest_len];
}
```

The Key Tag and Algorithm members of the DS record from RFC 4034 *[RFC4034]* section 5.1 are omitted because they are implied by the context. Additionally, the digest_len member may be zero to indicate that the digest itself is also implied by the context.

```
struct Name;
```

A Name is a series of 8-bit, unsigned, length-prefixed labels terminated by a the (empty) root label. Each label MUST be less than 64 bytes and the full name (including lengths and the root label) MUST be less than 256 bytes.

```
struct Entry {
  uint8 entry_key_index;
  struct Signature key_sig;
  uint8 num_keys;
  struct Key keys[num_keys];
}
```

The entry to a zone contains a number of keys and an optional signature over those keys. One of the keys (keys[entry_key_index]) is distinguished and called the entry key. The entry key is assumed to be trusted and bit 7 of the flags field MUST be set. The entry key may have zero length, in which case its value is that of the shared, initial key.
The entry key may sign the resource records for this zone, in which case key_sig.length SHOULD be zero, entry_key_index SHOULD be zero and num_keys SHOULD BE one. Otherwise, key_sig is a signature over keys, which MUST be in be in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3 after any empty keys have been substituted.

```
struct Exit {
  Name next_name;
  uint16 rrtype;
  struct Signature rrsig;

  select (rrtype) {
    case CNAME:
      Name cname;
    case DS:
      uint8 num_ds;
      struct Ds ds_records[num_ds];
    default:
      uint8 num_rrs;
      struct RRData rrs[num_rrs];
    }
  }
}
```

Leaving a zone involves a number of resource records at a domain in the zone. The resource records are signed by rrsig, which MUST be a valid

signature of them either from the entry key or, if key_sig.length is non-zero, one of the other keys.
All resource records MUST be in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3. If the digest member of any DS record is empty, then it's implied that it's the digest of the next zone's entry key and the order requirement is imposed after such substitution.
A serialized chain is a uint16 key tag of the shared, initial key followed by a series of Entry and Exit structures. These structures alternate except in the case a CNAME record, where upon several Exit structures can occur consecutively.

4. Verification

This text describes the process of verifying a serialized DNS record. Any process that produces a valid input to this process is a valid encoder.
The verification procedure starts in an initial state, advances to the ENTERING state and then cycles between that and a LEAVING state until a terminal LEAVING state, or error, is reached. A priori, the verifier is assumed to know the initial key and an initial zone. For the purposes of this text, the initial key is assumed to be the IANA root key and the initial zone is assumed to be the root zone. However, the same scheme could be used when starting in a subzone. At all times the verifier also holds a target name. The target name is known a priori but may be updated during the course of verification. The target name is the domain name for which a DNS record is being verified. The verifier also maintains a stack of zones. Each zone has a name and a set of trusted keys for that zone.

4.1. The INITIAL State

In the INITIAL state the verifier reads the following:

```
uint16 initial_key_tag;
```

The value of initial_key_tag is the key tag (see RFC 4034 *[RFC4034]*, Appendix B) of the initial key. This serves as a check that the verifier's initial key and initial zone match that which is being assumed. The verifier checks that the entry key of the next zone matches the initial key. (Understanding this requires reading the next section.)
The verifier moves to the ENTERING state.

4.2. The ENTERING State

In the ENTERING state the verifier reads an Entry structure.
The verifier extracts the entry key, which is found in the Key array and indexed by entry_key_index. The entry key in the initial zone may have a zero length, in which case it's assumed to be the initial key.

Otherwise the rdata member of the Key structures MUST have the format for DNSKEY RRDATA described in RFC 4034 *[RFC4034]* section 2.1. The verifier MUST check that bit 7 of the DNSKEY flags is set.
As a precondition of the ENTERING state, the entry key has already been validated so the verifier adds the entry key to the set of trusted keys for the current zone.
If key_sig is non-empty then the verifier constructs an RRSIG RDATA from the information in the Signature structure and the current zone name. The verifier constructs an RRSET from the rdata member of the Key structures. (The Key structures MUST be in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3.)
If the constructed RRSIG is a valid signature over the constructed RRSET using the entry key, then all the keys from the Key array are added to the set of trusted keys for the current zone. If key_sig is non-empty and the signature is invalid then the verification fails.
The verifier moves to the LEAVING state.

4.3. The LEAVING State

In the LEAVING state the verifier reads a Exit structure;
If rrtype is DS then the verifier fills in any empty digests based on the entry key of the next Entry structure. (The verifier has to read ahead in order to calculate this.) If the verifier cannot calculate any digests then the verification fails. The verifier MUST check that at least one DS record is either filled in, or is a valid digest of the entry key of the next Entry structure. After substituting any empty digests, the Ds structures MUST be in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3.
The verifier checks that an RRSIG RDATA constructed from rrsig is a valid signature over the RRDATAs. If not, the verification fails. The RRDATAs MUST be in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3.
If rrtype is DS then the verifier MUST check that next_name matches a greater number labels, right-to-left, of the current target name than the current zone name. The verifier pushes the current zone on the stack and moves to the ENTERING state with next_name as the zone name.
If rrtype is CNAME then the verifier MUST check that next_name matches the current target name. The contents of the CNAME record then become the new target name. The verifier moves to the LEAVING state and pops the minimum number of zones from the stack so the the new target name is within the current zone.
If rrtype is NSEC or NSEC3 then the verifier MUST check that next_name is within the current zone. The verifier terminates successfully and returns the rrtype and RRDATA of the record. The record may be used to establish the non-existence of the target record but that isn't specified in this text.
Otherwise, the verifier MUST check that next_name matches the target name. The verifier terminates successfully and returns the rrtype and RRDATA of the record.

5. Construction

This section describes a possible procedure for building data structures from which a serialization can be easily marshaled. It's assumed that the goal is to serialize a DNS record of some type at a given domain in the public DNS.

First, make a request for a CNAME record at the domain. If a CNAME record exists, then it's the terminal record. Otherwise resolve a record of a desired type at the domain. If neither record exists, fail.

Now find the chain of zone names leading to that domain. The parent zone of a domain is typically returned as an SOA record in the authority section for any query to that domain, unless the domain is a zone cut itself. In the event that a domain has an SOA record, then simply remove the left-most label from the name and try again.

Construct an array of zone records, one for each zone leading to the domain name, including the root zone. For each zone, fetch the set of DNSKEY records (with DO flag to get the RRSIGs) and, for every zone expect the last one, fetch the DS records for the next zone (also with DO). If any zones don't have either of those record types then there isn't a DNSSEC chain to the desired record and the construction fails.

The DS records are the exit records for all zones save the last. For the last zone, the exit record is the terminal record. Sort all records so that they are in canonical order as specified in RFC 4034 *[RFC4034]* section 6.3.

For each zone, verify the signature on the exit records with each possible key (using key tags to skip some keys). The set of keys that sign the exit records are called exit keys.

For the root zone, the entry key is the shared root key. For all other zones, look at the parent zone's DS records and mark keys as possible entry keys if there's a DS record for them and the hash function is commonly accepted. (At the time of writing, SHA1 and SHA256 are such hash functions.) Filter the possible entry keys to only include those that have the SEP bit set in the flags. If any zone has no possible entry keys then the construction fails.

For each zone, take the intersection of possible entry keys and exit keys. If the intersection is non-empty, pick one of those keys and discard the rest along with the signature over the keys. Otherwise, filter the acceptable entry keys by considering only those for which a valid signature over the DNSKEY RRSET can be found. Pick an acceptable entry and exit key.

Serialize the current zones. If the target record is a CNAME then repeat this procedure for the new name. Before serializing the zones, find the most junior zone that is shared with the previous set of zones. Discard all zones senior to it and omit its Entry structure when serializing.

6. Deficiencies

This text does not deal with DNAME records *[RFC2672]*, unless the DNAME records are used to synthesize CNAME records.

7. Test vectors

The following serialization involves assumes the the initial zone is the root and that the initial key is the IANA root key (key tag 19036). The initial key is given here for completeness:

```
00000000  01 01 03 08 03 01 00 01  a8 00 20 a9 55 66 ba 42
00000010  e8 86 bb 80 4c da 84 e4  7e f5 6d bd 7a ec 61 26
00000020  15 55 2c ec 90 6d 21 16  d0 ef 20 70 28 c5 15 54
00000030  14 4d fe af e7 c7 cb 8f  00 5d d1 82 34 13 3a c0
00000040  71 0a 81 18 2c e1 fd 14  ad 22 83 bc 83 43 5f 9d
00000050  f2 f6 31 32 51 93 1a 17  6d f0 da 51 e5 4f 42 e6
00000060  04 86 0d fb 35 95 80 25  0f 55 9c c5 43 c4 ff d5
00000070  1c be 3d e8 cf d0 67 19  23 7f 9f c4 7e e7 29 da
00000080  06 83 5f a4 52 e8 25 e9  a1 8e bc 2e cb cf 56 34
00000090  74 65 2c 33 cf 56 a9 03  3b cd f5 d9 73 12 17 97
000000a0  ec 80 89 04 1b 6e 03 a1  b7 2d 0a 73 5b 98 4e 03
000000b0  68 73 09 33 23 24 f2 7c  2d ba 85 e9 db 15 e8 3a
000000c0  01 43 38 2e 97 4b 06 21  c1 8e 62 5e ce c9 07 57
000000d0  7d 9e 7b ad e9 52 41 a8  1e bb e8 a9 01 d4 d3 27
000000e0  6e 40 b1 14 c0 a2 e6 fc  38 d1 9c 2e 6a ab 02 64
000000f0  4b 28 13 f5 75 fc 21 60  1e 0d ee 49 cd 9e e9 6a
00000100  43 10 3e 52 4d 62 87 3d
```

The target name for this test vector is www.dnssec-exp.org. The serialization establishes that www.dnssec-exp.org is a CNAME for dnssec-exp.org and then results in a TXT record at that name. Since DNSSEC signatures expire, time based validation needs to be disabled for this test vector.

```
00000000  4a 5c 01 01 10 08 00 00  01 51 80 4c 8e ba ff 4c
00000010  7a f4 80 4a 5c 6d e6 a7  aa 97 ca 48 4a 49 bb ff
00000020  91 1a d8 c2 55 0d 80 1c  83 41 2b 89 70 87 d9 3a
00000030  23 9f dc a4 b6 0c e3 3a  5a be fc 2b cb b7 14 bd
00000040  bf 43 aa 03 34 e8 84 18  2e 95 0e 77 b1 e2 74 86
00000050  d8 c3 9c dc 00 80 8c 92  fc 92 fe 25 e3 6d 94 e5
00000060  c4 08 18 f7 c2 96 de 8c  85 73 22 0d 03 7a e0 a9
00000070  51 27 24 bc f8 bf 63 52  ef bd b8 07 e8 f6 aa 3d
00000080  02 28 fe 44 d4 e4 04 10  61 95 6c 87 06 2a 5c db
00000090  67 0d ac 1a 02 2e 66 35  08 28 29 24 8e ad 4c 3b
000000a0  6a 99 88 d6 32 a6 7d c3  0c 3d bc 0b ff 86 8d c6
000000b0  50 8b ad 1f 1f f7 06 c5  9e 0c 6f bb 3c 02 5c 67
000000c0  a7 af 53 63 5a ae 99 47  93 61 c8 4f fb 03 ec fe
000000d0  a3 c1 ee 45 f8 56 73 88  f4 14 6d 96 7c dc 88 01
000000e0  99 04 18 3d 42 56 57 54  d7 ed f8 6b 46 ff 01 e6
000000f0  1a 75 37 d4 f6 b2 57 61  6b bf 24 99 6d cd 63 c6
00000100  45 d0 0a 93 5c bb d3 36  ac fa 57 51 a0 cf 32 0e
00000110  b3 57 f3 02 77 02 00 88  01 00 03 08 03 01 00 01
00000120  bd 60 70 38 41 94 7f fd  32 58 14 c5 2d 22 93 67
00000130  70 63 f2 62 04 8a 55 f8  48 4a 65 5e cb 71 cc 4d
00000140  73 a4 25 8f 8e bb 42 31  d0 dc 58 26 da 2d 8b 13
00000150  dc 3a 2e a3 c5 0d 29 e0  e6 a5 6a d4 e6 05 c9 30
00000160  6d dc 34 29 a6 a0 e7 00  fa e2 4f 05 b9 a8 84 0d
00000170  0c d1 6f df 8c a8 eb 7b  00 74 17 94 e0 6f 6d bf
00000180  b7 73 4f 9b 0f c8 08 26  56 1e 47 0c 27 be e9 73
00000190  36 f8 87 a5 d7 e9 14 28  27 a5 f0 87 32 d7 d8 51
000001a0  00 00 03 6f 72 67 00 00  2b 00 90 08 01 00 02 a3
000001b0  00 4c 90 0c 80 4c 86 c3  f0 a1 20 10 1a ab 3c 50
000001c0  ae 3f 7f 49 4a 3f fc f1  fb 5c 63 76 7f 60 f8 0e
000001d0  08 30 23 b1 e7 a9 bc 23  a4 1a af c5 99 3e 85 1f
000001e0  0b f3 7f 04 12 5b 7e 26  b1 87 a0 4c d9 af f0 30
000001f0  eb 88 b0 f6 88 b5 10 b7  c9 a8 c8 c0 c0 26 69 16
00000200  90 ec ec 53 f2 c4 24 7c  24 c0 67 09 29 4b cc 80
00000210  92 e5 d9 c4 a3 18 0b 16  65 c2 11 7a 3b b1 c0 af
00000220  0b 93 e6 7b 76 25 18 7a  1e 8e 4f 50 f2 b4 da 72
00000230  72 44 3a 18 f9 ed 72 05  19 77 34 02 01 00 00 02
00000240  00 00 02 01 10 07 01 00  00 03 84 4c 90 ea 0d 4c
00000250  7e 66 fd 53 76 37 11 28  49 49 a9 aa 7e e7 c6 2b
00000260  0f 0e 76 b7 a6 15 4d cf  95 9e 5b 25 5a 52 cc 62
00000270  d7 31 2b 6a 35 f7 9f 5b  c1 5e ba c1 53 98 3e c8
00000280  6c 21 6f f1 93 b6 b7 6e  65 04 30 e2 0c 6d a5 dd
00000290  15 d4 01 b8 d3 9b d2 86  38 24 6f ed 03 47 b4 9e
000002a0  0b 1a e3 16 7d 20 68 50  e5 b1 2c ae 14 c0 dc 09
000002b0  31 6d a0 4e 55 dc 65 be  e5 89 e4 35 57 3e 2b da
000002c0  06 8d ef c8 df f9 f6 c3  09 39 c7 83 e9 e0 f0 2e
000002d0  ad 21 56 8b 60 f9 84 53  ac 1e 84 42 7a fa a1 bd
000002e0  86 61 12 d0 70 dc 54 50  0e be 1a 47 e4 38 96 b4
000002f0  e7 3c 26 ae 1d 7e 37 28  f5 54 e6 94 63 30 42 7e
00000300  52 b2 8b da 96 44 0c a4  da 33 28 77 02 df cf a0
```

```
00000310  c7 14 2b 68 bb 5d 1b 7e  32 5c f2 0e cb 1b e1 6d
00000320  1c b8 96 bf 0d 1a cc 0f  2f fd 25 ff 33 3d 89 6d
00000330  27 9e e0 b9 5a 72 bb 2f  e5 95 bb 40 b6 4c 11 6c
00000340  80 b6 9e a9 d9 31 61 b9  69 9c f2 e8 c5 a0 fd 07
00000350  59 87 38 ff 25 04 00 88  01 00 03 07 03 01 00 01
00000360  80 97 87 f6 40 06 2f 24  88 92 03 5d 89 b2 52 51
00000370  f3 0b 40 87 78 1c ea 72  9c 99 00 88 c2 ed d2 b5
00000380  c2 44 58 55 c5 2f 22 5a  53 3a 99 ce 55 57 dc 0b
00000390  73 f2 f5 48 bf c7 8e 6a  29 bd 0b ca db ca ed 66
000003a0  00 7b 75 b2 38 ec 24 e6  49 70 22 a4 42 ff 4a 78
000003b0  96 e6 9f 6d dd b2 85 13  05 ee ab 8e 05 5a 98 ac
000003c0  ba 07 c2 ff 22 f4 ba d5  fa bf 1d 84 1e eb 5e ff
000003d0  e5 91 34 88 ea 61 19 b2  0e 6b 0d f7 9e f1 8c b5
000003e0  00 88 01 00 03 07 03 01  00 01 c9 06 00 53 45 4a
000003f0  0b b8 e2 b0 4e 29 c8 19  b4 a3 63 27 e2 cd c7 c7
00000400  6d 60 31 eb c0 82 5f 44  14 96 60 4e c8 62 f4 cc
00000410  b9 99 7a 19 f0 af 34 d9  63 ca 40 e3 7b b6 bc fa
00000420  40 08 1d e7 c3 a4 d2 73  3a 32 f2 a4 4c 3c 4f d6
00000430  52 52 c8 6d a5 f6 d9 4d  0c fd b4 93 8b 61 72 db
00000440  6e 5f 2d d9 2d ab 18 2f  87 2d bf 8d 42 37 93 41
00000450  18 f6 93 97 da 27 31 dc  da ec 21 16 61 e1 e0 7a
00000460  53 26 82 c7 62 99 18 81  6a 65 01 08 01 01 03 07
00000470  03 01 00 01 8a 58 7e 3d  da 69 1c f3 93 15 90 a8
00000480  c7 65 ed 81 31 63 cd 4d  75 84 af fa a6 b2 b9 90
00000490  e8 76 76 7d 20 c8 74 6f  03 1c 61 a5 54 77 33 40
000004a0  6d 57 89 f2 07 7a 8e ad  6c 47 75 6f 3f f4 91 df
000004b0  a9 a6 1a cb 1b 57 85 1d  97 93 91 0e da a2 64 fd
000004c0  93 0c d0 c7 c4 49 ca 29  35 fe 8d 67 f2 b5 97 93
000004d0  ed dd c0 6d 2c c1 28 2d  2f ee e6 6b 33 a3 36 7a
000004e0  82 67 97 a8 9d eb aa c4  52 64 02 da 9f 43 ae b0
000004f0  e0 f4 5e ad 5c 2f 42 0f  fc c2 ef fc be 04 d3 69
00000500  88 e7 67 33 90 d7 93 b1  e1 66 6e eb 6b d1 3b 96
00000510  d2 f5 de 1d a6 c7 b9 04  81 4f 1e ea 7a 49 94 2a
00000520  17 8e b6 88 06 05 03 b6  16 2c e3 c5 bf b1 d4 c3
00000530  2e ee cd e7 da e3 08 6f  9b a6 29 7e 73 ca 19 f5
00000540  fe cd 47 7a a6 49 3a 53  3f 59 bc e9 1a 94 42 75
00000550  44 ae 27 eb 1f c2 a3 0e  a2 fe df 0c d4 74 5e 40
00000560  0a 46 30 b7 55 e1 3d 53  d4 fb 04 88 97 36 da 01
00000570  03 78 f4 f5 01 08 01 01  03 07 03 01 00 01 94 e3
00000580  6c 83 b9 90 8a 71 59 4b  72 5d cf 1a be c2 b2 1c
00000590  82 19 f8 b8 c2 d8 3b fc  9d a3 be 4f 3e 97 d9 fa
000005a0  b3 0c 2d 5b 76 ae c7 95  9c 2d 91 aa 93 90 c5 55
000005b0  27 ef 20 13 d1 48 ad e1  89 e1 cf 06 d4 67 5b 8d
000005c0  08 1b 3f 53 b2 60 81 bb  38 74 dc e2 1b f9 4f 63
000005d0  65 c9 6a fa 93 a4 05 cf  df 10 e3 3c 05 20 64 c5
000005e0  56 fc 01 86 6a cc 0d 8b  0e 4e d5 da 90 ae 90 c0
000005f0  7a 2f 03 5f bc dc 1b 14  00 2c 65 89 da 70 07 48
00000600  50 69 c6 c3 eb 1f 88 d9  10 83 cd 8b 93 ce 3e b8
00000610  26 fd 3f f5 7b 17 e8 06  15 dd e6 dc 82 7e 21 2f
```

```
00000620  58 c8 47 67 89 63 25 e5  ac 0a 16 c5 dc f1 71 6f
00000630  ff e7 27 8b e5 15 56 ba  14 71 7a 39 a7 49 59 c2
00000640  bb 19 1f 4b 80 10 e3 ce  4a 1f 6b 69 75 b5 9c 0a
00000650  8a 4b 25 9b 3a b7 0f 2a  de 35 9c a5 31 b3 76 1f
00000660  ef df 17 58 7c da 50 35  c3 c8 98 59 71 02 e9 f7
00000670  06 d3 91 3c 0d ab f2 d8  ba 30 da 09 10 75 0a 64
00000680  6e 73 73 65 63 2d 65 78  70 03 6f 72 67 00 00 2b
00000690  00 90 07 02 00 01 51 80  4c 90 ea 0d 4c 7e 66 fd
000006a0  93 b4 39 ee 27 29 54 9e  57 41 12 60 19 f6 3f a6
000006b0  ba d6 41 98 57 ec 30 9e  96 08 8c 13 a9 76 95 74
000006c0  cd cd 2e a6 22 21 44 3f  13 df 7a 33 f1 8c 4c f9
000006d0  a3 6d 50 38 fa 71 7b 7a  fe 54 a9 44 81 8c d5 04
000006e0  9e 46 89 da 26 43 40 f6  d7 23 48 07 0e 48 2e 19
000006f0  0d 41 27 85 75 1e a0 a9  ad 39 af 8d c1 ed c5 93
00000700  73 05 09 7a 8f f0 97 c3  98 ab cc 7c bf 48 ef ea
00000710  34 f3 e6 5d 8d 1b be 43  97 56 4e 60 9f dd 1b f5
00000720  15 6c 01 02 00 00 00 00  00 01 00 a8 01 00 03 05
00000730  03 01 00 01 d0 0a 64 8e  b5 90 0b 75 c2 eb 52 5f
00000740  a4 cb eb 1d 77 8d 84 2c  ef b6 88 d4 7c 50 4c 52
00000750  7b 9d 37 31 36 b2 5b 6c  47 b4 21 80 61 46 fa 5b
00000760  44 50 91 9d f8 c1 78 00  78 29 fe e2 08 65 f8 c9
00000770  df 69 0b 59 6b f4 93 9f  8b 25 0b 6b 93 12 06 57
00000780  c8 04 9d 3f 33 bd 2b 35  54 b9 98 75 e4 b0 49 3c
00000790  29 c1 fb 74 bc 91 82 9e  c5 61 a6 16 e1 f0 8f e9
000007a0  e1 23 55 5e fb f1 cc 8a  75 5d f2 86 89 0a 29 cb
000007b0  ca 33 de 9d 74 43 0d de  67 de 71 0f 7f a3 bb 22
000007c0  82 81 66 d1 bd 49 0d 89  45 d7 18 cf 98 2c 19 af
000007d0  63 16 20 91 03 77 77 77  0a 64 6e 73 73 65 63 2d
000007e0  65 78 70 03 6f 72 67 00  00 05 00 b0 05 03 00 00
000007f0  00 3c 4c ae eb fc 4c 87  59 f6 83 7f 01 90 de 40
00000800  a3 42 73 59 b7 c7 3a e1  01 ae 5c 0f d4 22 9d b5
00000810  88 b8 bb 14 95 7f fa a7  ff 48 30 90 4c 1b 71 33
00000820  1c 36 bc af ae 1d 97 ec  ee 71 b8 b1 2a 19 d7 f3
00000830  25 78 94 59 54 bb 2b ac  e6 3c 3e 7e 43 7c 2d c0
00000840  72 58 e8 4a a2 ed 1f 39  5f 6a 98 62 d0 fc 6e be
00000850  01 1f a0 39 f2 b4 4d 33  9b 4d e3 81 65 77 80 3d
00000860  2c 44 ee 1a 65 f7 25 68  95 3f a1 22 d5 9d 59 7e
00000870  a7 a6 f2 7e 4d 47 8b 98  4a 0d 16 a1 78 12 c7 90
00000880  e2 e9 6d e6 89 f3 a1 c4  30 bd e8 f9 75 c6 49 9e
00000890  5f 9c 1e 23 b5 0e 82 70  8e b3 0d 1c 0a 64 6e 73
000008a0  73 65 63 2d 65 78 70 03  6f 72 67 00 0a 64 6e 73
000008b0  73 65 63 2d 65 78 70 03  6f 72 67 00 00 10 00 b0
000008c0  05 02 00 00 00 3c 4c ae  f6 e5 4c 87 5e d9 83 7f
000008d0  2a 88 85 b8 f1 73 93 e3  85 ee c8 93 34 2d d2 47
000008e0  2d 86 e5 c6 18 69 94 94  7a 2d 7e a2 57 f1 75 3e
000008f0  51 bf aa ed dd f8 1a b9  1e 86 6f 7f e9 dc ad 83
00000900  da 7d cd 2d 4d 9b e3 66  92 b9 21 96 bf ea 70 55
00000910  bf b3 b0 85 90 72 5f 09  d1 7c 49 25 07 79 34 48
00000920  e7 1c aa 53 33 cb 62 b9  9f 42 6e 36 00 31 ab 9f
```

```
00000930  cc f5 a9 02 2f 75 4c f1  d6 86 3a 7f 08 e4 72 7d
00000940  34 e0 40 76 d1 f0 e3 98  37 e7 93 94 57 95 6c de
00000950  9e 17 57 3a 78 e9 8b 57  7f 74 e2 af f8 af 11 64
00000960  a9 e9 db b2 58 da 7c fe  3f 52 06 39 fc dd 1c 1c
00000970  01 00 3a 39 76 3d 74 6c  73 31 20 68 61 3d 73 68
00000980  61 31 20 68 3d 31 30 39  63 38 31 34 36 33 30 34
00000990  64 65 37 63 34 63 38 37  30 35 62 63 63 31 64 38
000009a0  61 32 63 32 33 37 66 30  35 35 38 64 37
```

8. References

[RFC1035]	Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
[RFC2672]	Crawford, M., "Non-Terminal DNS Name Redirection", RFC 2672, August 1999.
[RFC4033]	Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
[RFC4034]	Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

Appendix A. Changes

To be removed by RFC Editor before publication

Author's Address

Adam Langley Langley Google Inc EMail: agl@google.com