DNS Extensions
Internet-Draft
Expires: August 2, 2002

R. Arends
Nominum
M. Larson
VeriSign
D. Massey
USC/ISI
S. Rose
NIST

February 2002

Resource Records for DNS Security Extensions
draft-ietf-dnsext-dnssec-records-01

Status of this Memo

Copyright Notice

Abstract

The DNS Security Extensions (DNSSEC) introduce four resource records: the KEY, DS, SIG, and NXT resource records. This document defines the purpose and the RDATA format for each of these records. This document is part of a family of documents that describe the DNS Security Extensions (DNSSEC). The DNS Security Extensions are a collection of new resource records and protocol modifications that

provide source authentication for the DNS. This document obsoletes RFC 2535 and incorporates changes from all updates to RFC 2535.

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 [4].

Table of Contents

1. Introduction

The reader to assumed to be familiar with common DNSSEC terminology as defined in [13] and familiar with the basic DNS concepts described in RFC1034 [1] and RFC1035 [2].

The DNS Security Extensions (DNSSEC) introduce four resource records: KEY, DS, SIG, and NXT resource records. This document defines the purpose of each resource record, the RDATA format, the ASCII representation, and an example of each RR type is given. Sections 2-5 describe the KEY, DS, SIG, and NXT records. Section 6 describe the DNSSEC header bits.

1.1 DNSSEC Document Family

This document is part of a family of documents that define the DNS security extensions. The DNS security extensions (DNSSEC) are a collection of resource records and DNS protocol modifications that add source authentication the Domain Name System (DNS). An introduction to DNSSEC and definition of common terms can be found in (RFC TBA). A description of DNS protocol modifications can be found in (RFC TBA). This document defines the DNSSEC resource records.

2. The Key Resource Record

Public keys used by the DNS infrastructure are stored in KEY resource records. A secure DNS zone will store its public key in a KEY RR and this KEY RR can be used to authenticate other RR sets in the zone. The KEY RR MAY also be used to store other types of DNS public keys, such as the keys used by SIG(0) [10] or TKEY [9]. These public keys are used to authenticate DNS messages such as a request to dynamically update a DNS zone.

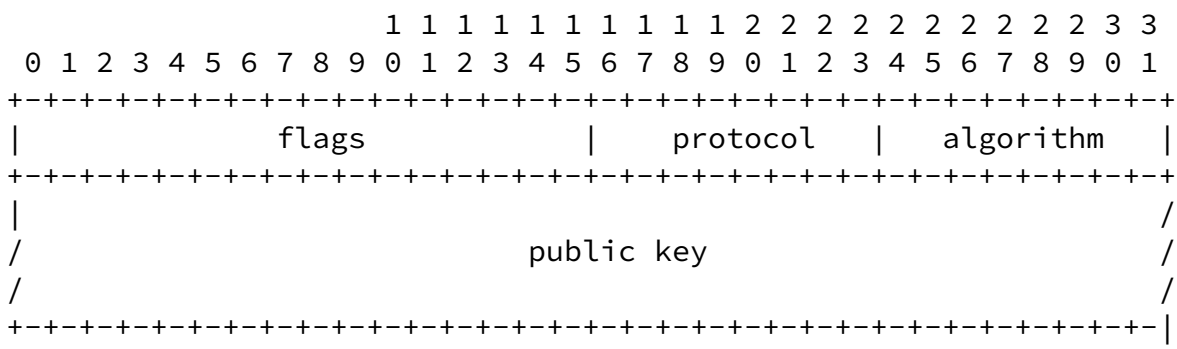
The KEY RR MUST only be used for public keys used for DNS purposes, all other uses are obsolete. The KEY RR plays an essential role in the secure processing of DNS messages and is included in various responses. The KEY RR MUST NOT be used to store certificates or public keys that do not directly relate to the DNS infrastructure. Examples of certificates and public keys that MUST NOT be stored in the KEY RR include X.509 certificates, IPSEC public keys, and SSH public keys.

The type number for the KEY RR is 25.

The KEY RR is class independent.

2.1 KEY RDATA Wire Format

The RDATA for a KEY RR consists of a 2 octet Flags Fields, a Protocol Octet, a one octet Algorithm number, and the public key.



2.1.1 The Flags Field

Bit 7 of the Flags Field is the "zone key flag". Bits 0-6 and 8-15 are reserved for future use. Bits 0-6 and 8-15 MUST be set to 0 and MUST be ignored during processing.

The zone key flag (bit 7) determines whether the KEY holds a DNS zone key. If bit 7 is 1, then the KEY record holds a DNS zone key. If bit 7 is 0, then the KEY record holds some other type of DNSSEC

infrastructure public key, such as a public key used by SIG(0) or TKEY. Resolvers MUST check the zone key flag in order to determine if the KEY record holds a DNS zone key.

2.1.1.1 Explanation for Choice of Bit 7

The choice of bit 7 as the zone key flag was made in order to provide backwards compatibility with an earlier version of the KEY record. This earlier version was defined in [6] and [15] eliminated all flags except the bit 7 zone key flag.

2.1.2 The Protocol Octet Field

The Protocol Octet value MUST be 3.

2.1.2.1 Explanation for a Fixed Value Protocol Octet Field

The Protocol Octet field is included for backwards compatibility with an earlier version of the KEY record. This earlier version of the KEY record was defined in [6] and [15] restricted the possible Protocol Octet values to 3.

2.1.3 The Algorithm and Public Key Fields

The Algorithm Field identifies the public key's cryptographic algorithm and determines the format of the Public Key Field.

Algorithm values are defined in separate documents. The following table shows the currently defined Algorithm formats:

VALUE	Algorithm	RFC	STATUS
0	Reserved	-	-
1	RSA/MD5	RFC 2536	NOT RECOMMENDED
2	Diffie-Hellman	RFC 2539	OPTIONAL
3	DSA	RFC 2536	MANDATORY
4	elliptic curve	Work in Progress	
5	RSA/SHA1	RFC 3110	MANDATORY
6-251	available for assignment	-	
252	reserved	-	indirect keys
253	private	-	domain name
254	private	-	OID
255	reserved	-	-

It is expected that a signed zone will contain at least one KEY record with one of the MANDATORY algorithms. A DNS security aware resolver MUST implement all MANDATORY and SHOULD implement all OPTIONAL algorithms. Currently RSA/MD5 is NOT RECOMMENDED for zone signing, but it may be found in older DNS implementations.

Therefore, if may be useful for a security aware resolver to implement RSA/MD5 as well as RSA/SHA1.

Algorithm number 252 is reserved for indirect key format where the actual key material is elsewhere (non-DNS). This format will be defined in a separate document.

Algorithm numbers 253 and 254 are reserved for private use and will never be assigned a specific algorithm. For number 253, the public key area and the signature begin with a wire encoded domain name indicating the algorithm the key uses. Only local domain name compression is permitted. The remainder of the public key area is privately defined. For number 254, the public key area for the KEY RR and the signature begin with an unsigned length byte followed by a BER encoded Object Identifier (ISO OID) of that length. The OID indicates the private algorithm in use and the remainder of the area is whatever is required by that algorithm. Entities should only use domain names and OIDs they control to designate their private algorithms.

2.2 The KEY RR Presentation Format

A KEY RR may appear as a single line. The presentation format of the RDATA portion is as follows:

The Flag field is represented as an unsigned integer.

The Protocol Octet field is represented as the unsigned integer 3.

The Algorithm Field is represented as an unsigned integer or as mnemonic specified. The mnemonic is listed in the document defining the algorithm.

The Public Key Field is a Base 64 encoding of the Public Key Field.

2.3 KEY RR Examples

2.3.1 Example 1

The following KEY RR stores a DNS zone key for isi.edu.

```
isi.edu. 86400 IN KEY 256 3 5 ( AQPT0sh3WjVeRY3WqpBjtf
                                <snip of base64 encoded text>
                                xxDw==)
```

256 indicates the flags field has the zone key bit is set. 3 is the fixed Protocol Octet value. 5 indicates the public key algorithm is RSA/SHA1 RFC 3110]. The remaining text is base 64 encoding of the

public key and the format of the public key is defined in [12].

Resolvers might use this public key to authenticate signed RR sets such as the A RR set for www.isi.edu. The authentication process used by resolvers is described in [14].

2.3.2 Example 2

The following KEY RR stores a public key used by SIG(0)

```
ddnskey.isi.edu. 86400 IN KEY 0 3 3 ( AQPT0sh3WjVeRY3WqpBjtf
                                      <snip of base64 encoded text>
                                      xxDw==)
```

0 indicates the flags field does not have the zone key bit is not set. 3 is the fixed Protocol Octet value. 5 indicates the public key algorithm is DSA [7]. The remaining text is base 64 encoding of the public key and the format of the public key is defined in [7].

This public key can be used to sign dynamic DNS updates for the isi.edu zone. The process is for signing the dynamic DNS updates is described in [11].

3. The SIG Resource Record

The SIG or "signature" resource record (RR) is the fundamental way that data is authenticated in the secure Domain Name System (DNS). As such it is the heart of the security provided.

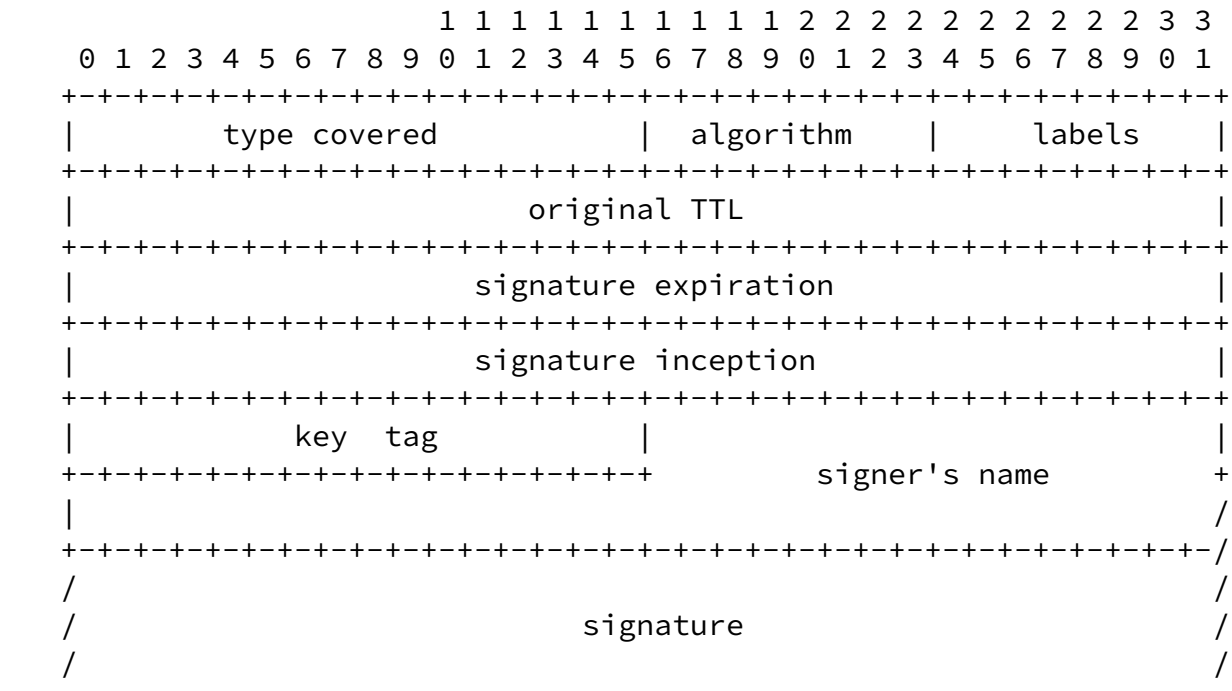
The SIG RR authenticates an RRset [5] of a particular type, class, and name and binds it to a time interval and the signer's name. The signer is the key (and associated KEY record) from which the RR originated. A SIG record can also be used for transaction security [transaction ref/section]. This type of SIG is known as SIG(0) and its RDATA is in the same format, with some values loosing their meaning and given default values. The variations are mentioned in [10].

The type number for the SIG RR type is 24.

The SIG RR is class independent, but MUST have the same class as the RRset it covers. The TTL for the SIG RR SHOULD be the same as the RRset it covers.

3.1 The SIG RDATA

The RDATA portion of a SIG RR is as shown below:



3.1.1 The Type Covered Field

For RRset SIGs, the type covered MUST be the same as the type of data in the associated RRset. For SIG(0), this field MUST be zero [10]

3.1.2 The Algorithm Number Field

The Algorithm Number field in the RDATA is the same field as found in the algorithm field of the KEY record RDATA [section 2.2.3].

3.1.3 The Labels Field

The "labels" octet is an unsigned count of how many labels there are in the original SIG RR owner name. This does not count null labels for root and any initial "*" for a wildcard. The labels count MUST be less than or equal to the number of labels in the SIG owner name.

3.1.4 Original TTL Field

The "original TTL" field is included in the RDATA portion to avoid authentication problems caused by caching servers decrementing the real TTL field. The signatures covers this field (as part of the SIG RDATA) while the TTL field is not. In a SIG(0), the Original TTL field (and the TTL field) MUST be zero.

The "original TTL" value MUST be greater than or equal to the TTL of the SIG record itself.

3.1.5 Signature Expiration and Inception Fields

The SIG is valid from the "signature inception" time until the "signature expiration" time. Both are unsigned numbers of seconds since the start of 1 January 1970, GMT, ignoring leap seconds. Ring arithmetic is used as for DNS SOA serial numbers [3], which means that these times can never be more than about 68 years in the past or the future. This means that these times are ambiguous modulo ~136.09

years.

A SIG RR may have an expiration time numerically less than the inception time if the expiration time is near the 32-bit wrap around point and/or the signature is long lived.

3.1.6 The Key Tag Field

The "Key Tag" is a two-octet quantity that is used to efficiently select between multiple keys that may be applicable. The Key Tag value may differ depending on the key algorithm in use, as described in Appendix (A).

3.1.7 The Signer's Name Field

The signer's name field MUST contain the name of the zone to which the data and signature belong. The combination of signer's name, key tag, and algorithm MUST identify a zone key if the SIG is to be considered material. In a SIG(0), the signer's name MUST be the originating host of the DNS message [10].

3.1.8 The Signature Field

The actual signature portion of the SIG RR binds the other RDATA fields to the RRset of the "type covered" RRs with that owner name and class.

3.2 The NXT RR Presentation Format (placeholder)

This section will be here in the next revision.

3.3 Calculating the signature

To generate the signature over an RRset, a data sequence is constructed as follows (where "|" is concatenation):

```
signature = sign(RDATA | RR(1) | RR(2)...  )

RR(N) = name | class | type | original TTL(stored in SIG RDATA) |
RDATA
```

To generate a signature over a DNS message (SIG(0)), a data sequence

is constructed as follows:

If the DNS message is sent via UDP:

   signature = sign(RDATA | full query | full response - SIG(0))

If the DNS message is sent via TCP, the first packet's SIG(0) is calculated as above, with each additional packet (if any) calculated as follows:

   signature = sign(RDATA | DNS payload - SIG(0) | previous packet)

where "previous packet" is the previous DNS packet with accompanying SIG(0), but without any other headers (i.e. TCP/IP, etc.).

In all the examples,

RDATA is the wire format of all the RDATA fields in the SIG RR itself (including the canonical form of the signer's name) before but not

including the signature, and

RR(num) is the RRset with the same owner name and class and type covered as the SIG RR in canonical form.

Name is the Fully Qualified Domain Name (FQDN) in canonical form.

The canonical form for a Resource Record (RR) is the wire format of the RR. Names MUST be expanded (no name compression allowed). Name characters MUST be set to lower case. Wildcards MUST be unexpanded. The RR MUST have the original TTL.

How this data sequence is processed into the signature is algorithm dependent. These algorithm dependent formats and procedures are described in separate documents.

SIGs SHOULD NOT be generated for any "meta-type" such as ANY, AXFR, etc.

4. The NXT Resource Record

The collection of NXT or "next" resource records (RR) is used to indicate what names and RRsets [5] exist in a zone.

The NXT RR lists the next canonical name in the zone and lists what RR types are present for the current name of the NXT RR.

The set of NXT RRs in a zone is a chain of all authoritative names in that zone.

Glue address records MUST NOT be covered by a NXT RR.

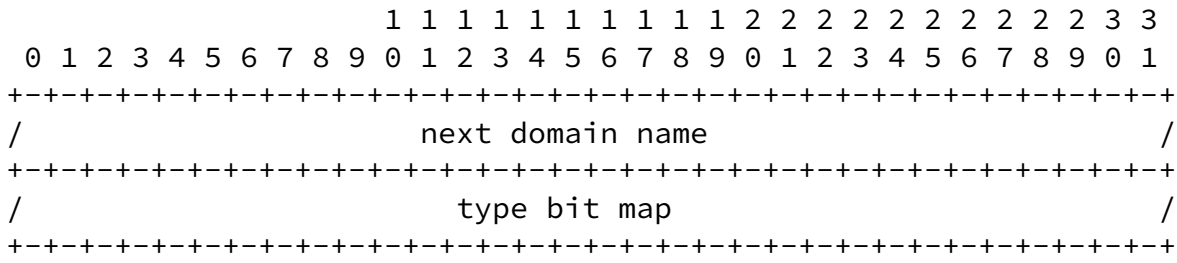
The type number for the NXT RR is 30.

The NXT RR is class independent.

The NXT RR TTL SHOULD NOT exceed the zone minimum TTL.

4.1 NXT RDATA Wire Format

The RDATA of the NXT RR is as shown below:



4.1.1 The Next Domain Name Field

The "next domain name" field contains the next owner name in canonical order. Canonical order means sorted by label, highest level label first. The "next domain name" field of the NXT RR at the last name in the zone contains the zone apex name.

Glue address record names MUST NOT be covered by the "next domain name" field.

The "next domain name" field allows message compression.

4.1.2 The Type Bit Map Field

The "type bit map" field format contains a single bit per RR type for RRsets with the same owner name as the NXT RR. A one bit indicates

that an RRset of that type exist for the owner name. A zero bit indicates that no RRset of that type exist for the owner name.

The first bit represents RR type zero. RR type number zero is not assigned and the corresponding bit MUST be zero. If the zero bit is one, it indicates that an unspecified format is used. This format is not used when there exist an RR type number greater than 127.

The OPT RR [8] type MUST NOT be covered by the type bit map field since it is not part of the zone data. The corresponding OPT RR type bit (40) MUST be zero.

Trailing zero octets MUST be omitted. Trailing zero octets not specified MUST be interpreted as zero octets. Glue address record types MUST NOT be covered by the type bit map field.

4.2 The NXT RR Presentation Format

A NXT RR may appear as a single line. The presentation format of the RDATA portion is as follows:

The "next domain name" field is represented as a domain name.

The "type bit map" field is represented as a sequence of RR type mnemonics or as an unsigned integer.

5. The DS Resource Record

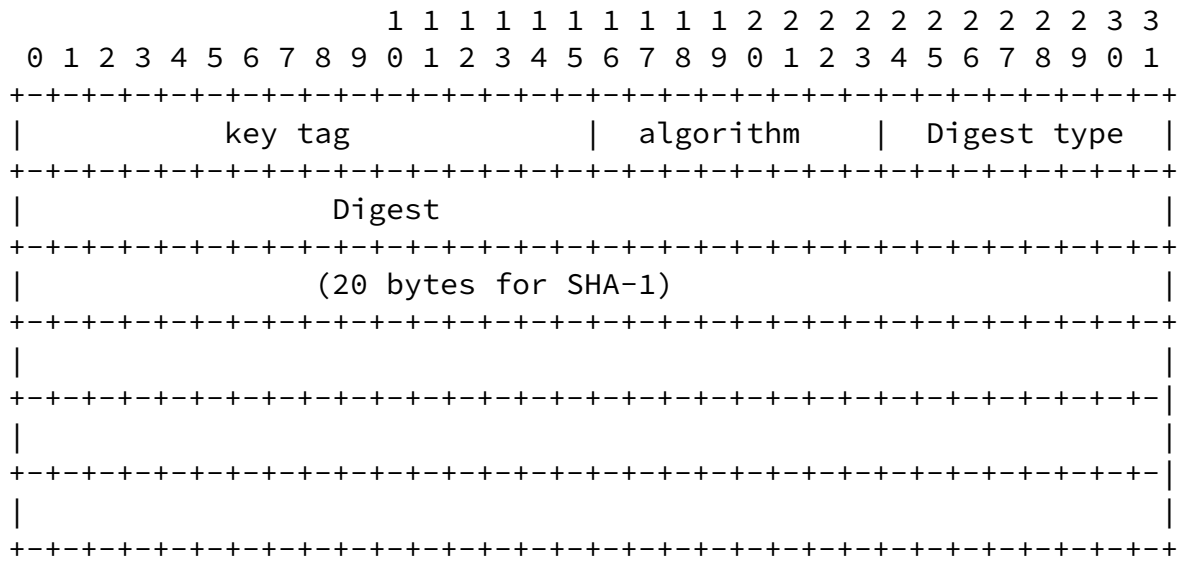
The DS record is a major change to DNS: it is the first resource record that can appear only on the upper side of a delegation. Other keys MAY sign the child's apex KEY RRset. DS records MUST point to zone KEY records that are allowed to authenticate DNS data.

The type number for the DS record is 43.

The DS record is class independent.

5.1 DS RDATA Wire Format

This record contains these fields: key tag, algorithm, digest type, and the digest of a public key KEY record that is allowed and/or used to sign the child's apex KEY RRset.



5.1.1 The Key Tag Field

The key tag value is the same key tag value in the SIG RRs generated using the KEY record this DS record points too. Having the key tag in the RDATA provides additional reliability in matching than just the KEY digest alone. See the key tag for details.

5.1.2 The Algorithm Field

The algorithm value has the same defined values as the KEY and SIG records. The value MUST be an algorithm number assigned in the range 1..251 and the algorithm MUST be allowed to sign DNS data.

5.1.3 The Digest Type Field

The digest type is an identifier for the digest algorithm used. The following numbers have been assigned and the assignment of future numbers requires IETF standards action.

VALUE	Algorithm	STATUS
0	Reserved	-
1	RSA/SHA-1	MANDATORY
2-255	Unassigned	-

5.1.4 The Digest Field

The digest is calculated over the canonical name of the delegated domain name followed by the whole RDATA of the KEY record (all four fields). The size of the DS RDATA for type 1 (SHA-1) is 24 bytes, regardless of key size. Other digest algorithms may have a differing digest size, to be described in other documents.

```
digest = hash( cannonical FQDN on KEY RR | KEY_RR_rdata)

KEY_RR_rdata = Flags | Protocol | Algorithm | Public Key
```

5.2 DS Record Example

The presentation format of the DS record consists of three numbers (key tag, algorithm and digest type) followed by the digest itself presented in hex:

```
example.   DS  12345 3 1 123456789abcdef67890123456789abcdef67890
```

This is a example of a KEY record and corresponding DS record.

```
dskey.example. KEY  256 3 1 (
                encoded public key
                ) ; key id = 28668
           DS   28668 1  1  49FD46E6C4B45C55D4AC69CBD3CD34AC1AFE51DE
```

5.3 Resolver Example

To create a chain of trust, a resolver goes from trusted KEY to DS to KEY.

Assume the key for domain "example." is trusted. Zone "example."


```
contains at least the following records:
example.           SOA     (soa stuff)
example.           NS       ns.example.
example.           KEY     (encoded public key)
example.           NXT      NS SOA KEY SIG NXT
example.           SIG(SOA)
example.           SIG(NS)
example.           SIG(NXT)
example.           SIG(KEY)
secure.example.    NS      ns1.secure.example.
secure.example.    DS      tag=10243 alg=3 digest_type=1
secure.example.    NXT     NS SIG NXT DS unsecure.example.
secure.example.    SIG(NXT)
secure.example.    SIG(DS)
unsecure.example   NS      ns1.unsecure.example.
unsecure.example.  NXT     NS SIG NXT .example.
unsecure.example.  SIG(NXT)

In zone "secure.example." following records exist:
secure.example.    SOA      (soa stuff)
secure.example.    NS       ns1.secure.example.
secure.example.    KEY      (tag=12345 alg=3)
secure.example.    SIG(KEY) (key-tag=12345 alg=3)
secure.example.    SIG(SOA) (key-tag=12345 alg=3)
secure.example.    SIG(NS)  (key-tag=12345 alg=5)
```

In this example the private key for "example." signs the DS record for "secure.example.", making that a secure delegation. The DS record states which key is expected to sign the RRsets at "secure.example.". Here "secure.example." signs its KEY RRset with the KEY identified in the DS RRset, thus the KEY RRset is validated and trusted.

This example has only one DS record for the child, but parents MUST allow multiple DS records to facilitate key rollover. It is strongly recommended that the DS RRset be kept small: two or three DS records should be sufficient in all cases.

The resolver determines the security status of "unsecure.example." by examining the parent zone's NXT record for this name. The absence of the DS bit indicates an unsecure delegation.

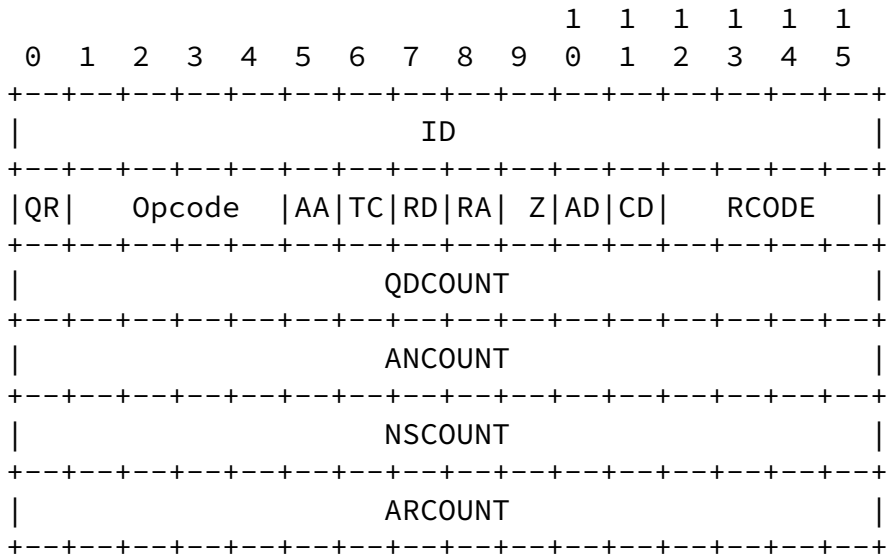
6. DNSSEC message bits

There are 3 new bits allocated for use with DNSSEC. The DO bit is used to indicate to a server that the resolver is able to accept DNSSEC security RRs (KEY SIG NXT DS). The CD and AD bits are used to indicate if non-authenticated data is accepted, and if data is authenticated.

6.1 The AD and CD Header Bits

Two bits are allocated in the header section. The CD (checking disabled) bit and the AD (authentic data) bit.

The Header contains the following fields:

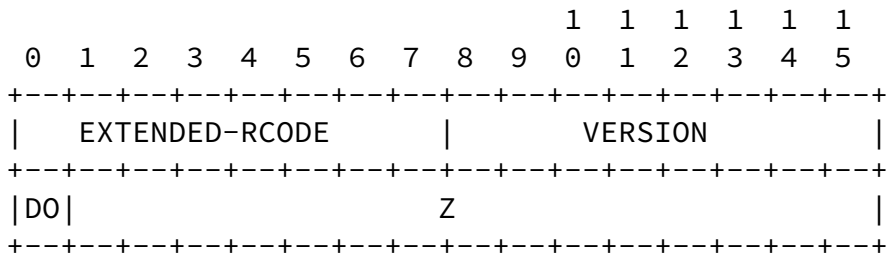


The usage of the CD and AD bits are defined in [14]

6.2 The DO Extended Flags Field Bit

The DO (DNSSEC OK) bit is allocated from the EDNS0 [8] extended flags field. In the context of the OPT RR, the DO bit is the most significant bit in the 3rd octet of the TTL field.

The TTL field of the OPT RR is defined as follows:



The usage of the DO bit is defined in [14]

7. IANA Considerations

This document clarifies the use of existing types and introduces no new IANA considerations.

The definitions of the flag bits in the KEY RR are set by working group consensus and there is no IANA registry for their definition. Changes to the meaning of the bits in the flags section of the KEY RDATA must be done through working group consensus.

RFC 2535 created an IANA registry for DNSSEC Resource Record algorithm Octet values. Values to 1-5, and 255 were assigned and values 6-254 were made available for assignment by IANA. This document re-assigns DNS KEY Resource Record Protocol Octet values 1, 2, 4, and 255 to ``reserved''. DNS Key Resource Record Protocol Octet Value 3 remains unchanged as ``DNSSEC''.

New protocol values are no longer available for assignment by IANA

and this document closes the IANA registry for DNS KEY Resource Record Protocol Octet Values. Assignment of any future KEY Resource Record Protocol Octet values requires a standards action. New numbers for algorithm values will continue to be assigned by IANA.

IANA needs to open a new registry for the DS RR type digest algorithms. Defined types are: 0 is Reserved, 1 is SHA-1. Adding new reservations requires IETF standards action.

8. Security Considerations

This document describes the format of resource records used by DNS security. The threats facing DNS are described in a separate document and these records are used to help counter those threats. The records themselves introduce no new security considerations, but the protocol use of these records is described in a second document.

9. Acknowledgements

This document was created from the input and ideas of several members of the DNS Extensions Working Group and working group mailing list.

The co-authors of this draft would like to express their thanks for the comments and suggestions received during the re-writing of these security extension specifications.

References

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

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

[3] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

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

[5] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[6] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[7] Eastlake, D., "DSA KEYs and SIGs in the Domain Name System (DNS)", RFC 2536, March 1999.

[8] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[9] Eastlake, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.

[10] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[11] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[12] Eastlake, D., "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name System (DNS)", RFC 3110, May 2001.

[13] Arends, R., Larson, M., Massey, D. and S. Rose, "DNSSEC Intro", February 2002.

[14] Arends, R., Larson, M., Massey, D. and S. Rose, "DNSSEC Protocol", February 2002.

[15] Massey, D. and S. Rose, "Limiting the Scope of the KEY Resource Record", draft-ietf-dnsext-restrict-key-for-dnssec-01 (work in progress), January 2002.

Authors' Addresses

Roy Arends
Nominum, Inc.
2385 Bay Street
Redwood City, CA 94063
USA

EMail: roy.arends@nominum.com

Matt Larson
VeriSign, Inc.
21345 Ridgetop Circle
Dulles, VA 20166-6503
USA

EMail: mlarson@verisign.com

Dan Massey
USC Information Sciences Institute
3811 N. Fairfax Drive
Arlington, VA 22203
USA

EMail: masseyd@isi.edu

Scott Rose
National Institute for Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-3460
USA

EMail: scott.rose@nist.gov

Appendix A. Key Tag Calculation

The key tag field in the SIG RR is just a means of more efficiently selecting the correct KEY RR to use when there is more than one KEY RR candidate available, for example, in verifying a signature. It is possible for more than one candidate key to have the same tag, in which case each must be tried until one works or all fail. The following reference implementation of how to calculate the Key Tag, for all algorithms other than algorithm 1 (which is NOT RECOMMENDED), is in ANSI C. The input is the key material in base 64,not the entire RDATA of the KEY record that contains the public key. It is coded for clarity, not efficiency.

```
/* assumes int is at least 16 bits
   first byte of the key tag is the most significant byte of return
   value
   second byte of the key tag is the least significant byte of
   return value
   */

int keytag (

        unsigned char key[],  /* the RDATA part of the KEY RR */
        unsigned int keysize, /* the RDLENGTH */
        )
{
long int    ac;    /* assumed to be 32 bits or larger */

for ( ac = 0, i = 0; i &lt keysize; ++i )
    ac += (i&amp1) ? key[i] : key[i]&lt&lt8;
ac += (ac>>16) &amp 0xFFFF;
return ac &amp 0xFFFF;
}
```

Full Copyright Statement

MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.