

DNS Extensions
Internet-Draft
Expires: August 26, 2003

R. Arends
Telematica Instituut
R. Austein
ISC
M. Larson
VeriSign
D. Massey
USC/ISI
S. Rose
NIST
February 25, 2003

Resource Records for the DNS Security Extensions
draft-ietf-dnsext-dnssec-records-03

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 26, 2003.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document is part of a family of documents that describes the DNS Security Extensions (DNSSEC). The DNS Security Extensions are a collection of resource records and protocol modifications that provide source authentication for the DNS. This document defines the

Internet-Draft

DNSSEC Resource Records

February 2003

KEY, DS, SIG, and NXT resource records. The purpose and format of each resource record is described in detail and an example of each resource record is given.

Table of Contents

1.	Introduction	4
1.1	Background and Related Documents	4
1.2	Reserved Words	4
1.3	Editors Notes	4
1.3.1	Open Technical Issues	4
1.3.2	Technical Changes or Corrections	4
1.3.3	Typos and Minor Corrections	5
2.	The KEY Resource Record	6
2.1	KEY RDATA Wire Format	6
2.1.1	The Flags Field	6
2.1.2	The Protocol Field	7
2.1.3	The Algorithm Field	7
2.1.4	The Public Key Field	7
2.1.5	Notes on KEY RDATA Design	7
2.2	The KEY RR Presentation Format	7
2.3	KEY RR Example	7
3.	The SIG Resource Record	9
3.1	SIG RDATA Wire Format	9
3.1.1	The Type Covered Field	10
3.1.2	The Algorithm Number Field	10
3.1.3	The Labels Field	10
3.1.4	Original TTL Field	11
3.1.5	Signature Expiration and Inception Fields	11
3.1.6	The Key Tag Field	11
3.1.7	The Signer's Name Field	11
3.1.8	The Signature Field	12
3.2	The SIG RR Presentation Format	12
3.3	SIG RR Example	13
4.	The NXT Resource Record	15
4.1	NXT RDATA Wire Format	15
4.1.1	The Next Domain Name Field	15
4.1.2	The Type Bit Map Field	15
4.1.3	Inclusion of Wildcard Names in NXT RDATA	16
4.2	The NXT RR Presentation Format	16
4.3	NXT RR Example	16
5.	The DS Resource Record	18

5.1	DS RDATA Wire Format	18
5.1.1	The Key Tag Field	18
5.1.2	The Algorithm Field	19
5.1.3	The Digest Type Field	19
5.1.4	The Digest Field	19
5.2	The DS RR Presentation Format	19

5.3	DS RR Example	20
6.	Canonical Form and Order of Resource Records	21
6.1	Canonical DNS Name Order	21
6.2	Canonical RR Form	21
6.3	Canonical RR Ordering Within An RRset	22
7.	IANA Considerations	23
8.	Security Considerations	24
9.	Acknowledgments	25
	Normative References	26
	Informative References	27
	Authors' Addresses	27
A.	DNSSEC Algorithm and Digest Types	29
A.1	DNSSEC Algorithm Types	29
A.1.1	Private Algorithm Types	29
A.2	DNSSEC Digest Types	30
B.	Key Tag Calculation	31
B.1	Key Tag for Algorithm 1 (RSA/MD5)	32
	Full Copyright Statement	33

[1.](#) Introduction

The DNS Security Extensions (DNSSEC) introduce four new DNS resource record types: KEY, SIG, NXT, and DS. This document defines the purpose of each resource record (RR), the RR's RDATA format, and its ASCII representation.

[1.1](#) Background and Related Documents

The reader is assumed to be familiar with the basic DNS concepts described in [RFC1034](#) [1] and [RFC1035](#) [2].

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 [\[10\]](#). A description of DNS protocol modifications can be found in [\[11\]](#). This document defines the DNSSEC resource records.

[1.2](#) Reserved Words

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](#) [5].

[1.3](#) Editors Notes

[1.3.1](#) Open Technical Issues

The NXT section ([Section 4](#)) may be updated in the next version if DNSSEC-Opt-In [[13](#)] becomes part of DNSSEC.

The cryptographic algorithm types (Appendix A) requires input from the working group. The DSA algorithm was moved to OPTIONAL. This had strong consensus in workshops and various discussions and a separate internet draft solely to move DSA from MANDATORY to OPTIONAL seemed excessive. This draft solicits input on that proposed change.

[1.3.2](#) Technical Changes or Corrections

Please report technical corrections to dnsssec-editors@east.isi.edu. To assist the editors, please indicate the text in error and point out the RFC that defines the correct behavior. For a technical change where no RFC that defines the correct behavior, or if there's more than one applicable RFC and the definitions conflict, please post the issue to [namedroppers](#).

Arends, et al.

Expires August 26, 2003

[Page 4]

Internet-Draft

DNSSEC Resource Records

February 2003

An example correction to [dnsssec-editors](mailto:dnsssec-editors@east.isi.edu) might be: Page X says "DNSSEC RRs SHOULD be automatically returned in responses." This was true in [RFC 2535](#), but [RFC 3225](#) ([Section 3](#), 3rd paragraph) says the DNSSEC RR types MUST NOT be included in responses unless the resolver indicated support for DNSSEC.

[1.3.3](#) Typos and Minor Corrections

Please report any typos corrections to dnsssec-editors@east.isi.edu. To assist the editors, please provide enough context for us to find the incorrect text quickly.

An example message to [dnsssec-editors](mailto:dnsssec-editors@east.isi.edu) might be: page X says "the DNSSEC standard has been in development for over 1 years". It should read "over 10 years".

[2](#). The KEY Resource Record

DNSSEC uses public key cryptography to sign and authenticate DNS resource record sets (RRsets). The public keys are stored in KEY resource records and are used in the DNSSEC authentication process described in [\[11\]](#). In a typical example, a zone signs its authoritative RRsets using a private key and stores the corresponding public key in a KEY RR. A resolver can then use these signatures to authenticate RRsets from the zone.

The KEY RR may also be used to store public keys associated with other DNS operations such as TKEY [\[15\]](#). In all cases, the KEY RR plays a special role in secure DNS resolution and DNS message processing. The KEY RR is not intended as a record for storing

arbitrary public keys. 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.

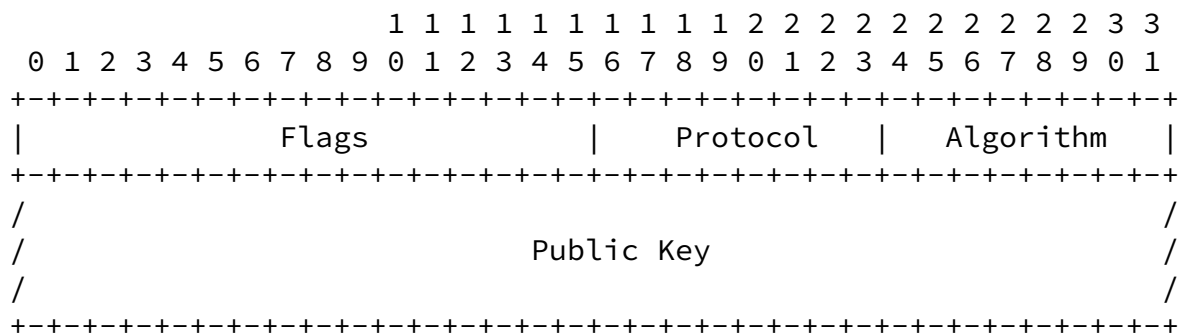
The Type value for the KEY RR type is 25.

The KEY RR is class independent.

There are no special TTL requirements on the KEY record.

2.1 KEY RDATA Wire Format

The RDATA for a KEY RR consists of a 2 octet Flags Field, a 1 octet Protocol Field, a 1 octet Algorithm Field , and the Public Key Field.



2.1.1 The Flags Field

Bit 7 of the Flags field is the Zone Key flag. If bit 7 has value 1, then the KEY record holds a DNS zone key and the KEY's owner name MUST be the name of a zone. If bit 7 has value 0, then the KEY record holds some other type of DNS public key, such as a public key

used by TKEY.

Bits 0-6 and 8-15 are reserved and MUST have value 0 upon creation of the KEY RR, and MUST be ignored upon reception.

Editors' Note: [draft-ietf-dnsext-keyrr-key-signing-flag](#) changes this by allocating bit 15 as the KSK bit.

[2.1.2](#) The Protocol Field

The Protocol Field MUST have value 3.

[2.1.3](#) The Algorithm Field

The Algorithm field identifies the public key's cryptographic algorithm and determines the format of the Public Key field. A list of DNSSEC algorithm types can be found in [Appendix A.1](#)

[2.1.4](#) The Public Key Field

The Public Key Field holds the public key material.

[2.1.5](#) Notes on KEY RDATA Design

Although the Protocol Field always has value 3, it is retained for backward compatibility with an earlier version of the KEY record.

[2.2](#) The KEY RR Presentation Format

The presentation format of the RDATA portion is as follows:

The Flag field is represented as an unsigned decimal integer with a value of either 0 or 256.

The Protocol Field is represented as an unsigned decimal integer with a value of 3.

The Algorithm field is represented either as an unsigned decimal integer or as an algorithm mnemonic as specified in [Appendix A.1](#).

The Public Key field is represented as a Base64 encoding of the Public Key. Whitespace is allowed within the Base64 text. For a definition of Base64 encoding, see [\[3\] Section 5.2](#).

[2.3](#) KEY RR Example

The following KEY RR stores a DNS zone key for example.com.


```
example.com. 86400 IN KEY 256 3 5 ( AQPSKmynfzW4kyBv015MUG2DeIQ3Cbl
+BBZH4b/0PY1kxkmvHjcZc8nokfzj31
GajIQKY+5CptLr3buXA10hWqTkF7H6R
foRqXQeogmMHfpftf6zMv1LyBUgia7z
a6ZEz0JB0ztyvhjL742iU/TpPSEDhm2
SNKLijfUppn1UaNvv4w== )
```

The first four text fields specify the owner name, TTL, Class, and RR type (KEY). Value 256 indicates that the Zone Key bit (bit 7) in the Flags field has value 1. Value 3 is the fixed Protocol value. Value 5 indicates the public key algorithm. [Appendix A.1](#) identifies algorithm type 5 as RSA/SHA1 and indicates that the format of the RSA/SHA1 public key field is defined in [8]. The remaining text is a base 64 encoding of the public key.

3. The SIG Resource Record

DNSSEC uses public key cryptography to sign and authenticate DNS resource record sets (RRsets). Signatures are stored in SIG resource records and are used in the DNSSEC authentication process described in [11]. In a typical example, a zone signs its authoritative RRsets using a private key and stores the corresponding signatures in SIG RRs. A resolver can then use these SIG RRs to authenticate RRsets from the zone.

A SIG record contains the signature for an RRset with a particular name, class, and type. The SIG RR specifies a validity interval for the signature and uses the Algorithm, the Signer's Name, and the Key Tag to identify the public key (KEY RR) that can be used to verify the signature.

The SIG RR may cover a transaction instead of an RRset. In this case, the "Type Covered" field value is 0, the SIG RR MUST NOT appear in any zone, and its use and processing are outside the scope of this document. Please see [7] for further details.

The Type value for the SIG RR type is 24.

The SIG RR MUST have the same class as the RRset it covers.

The SIG RR TTL value SHOULD match the TTL value of the RRset it covers.

3.1 SIG RDATA Wire Format

The RDATA for a SIG RR consists of a 2 octet Type Covered field, a 1 octet Algorithm field, a 1 octet Labels field, a 4 octet Original TTL field, a 4 octet Signature Expiration field, a 4 octet Signature Inception field, a 2 octet Key tag, the Signer's Name field, and the Signature field.

```

      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Type Covered           | Algorithm | Labels |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Original TTL                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Signature Expiration									
Signature Inception									
Key Tag									

Signer's Name									
Signature									

[3.1.1](#) The Type Covered Field

The Type Covered field identifies the type of the RRset which is covered by this SIG record.

If Type Covered field has a value of 0, the record is referred to as a transaction signature; please see [\[7\]](#) for further details.

[3.1.2](#) The Algorithm Number Field

The Algorithm Number field identifies the cryptographic algorithm used to create the signature. A list of DNSSEC algorithm types can be found in [Appendix A.1](#)

[3.1.3](#) The Labels Field

The Labels field specifies the number of labels in the original SIG RR owner name. It is included to handle signatures associated with wildcard owner names.

To validate a signature, the validator requires the original owner name that was used when the signature was created. If the original owner name contains a wildcard label ("*"), the owner name may have been expanded by the server during the response process, in which case the validator will need to reconstruct the original owner name in order to validate the signature. [\[11\]](#) describes how to use the Labels field to reconstruct the original owner name.

The value of the Label field MUST NOT count either the null (root) label that terminates the owner name or the wildcard label (if present). The value of the Label field MUST be less than or equal to the number of labels in the SIG owner name. For example, "www.example.com." has a Label field value of 3, and "*.example.com." has a Label field value of 2. Root (".") has a Label field value of 0.

Note that, although the wildcard label is not included in the count stored in the Label field of the SIG RR, the wildcard label is part of the RRset's owner name when generating or verifying the signature.

[3.1.4](#) Original TTL Field

The Original TTL field specifies the TTL of the covered RRset as it appears in the authoritative zone.

The Original TTL field is necessary because a caching resolver decrements the TTL value of a cached RRset. In order to validate a signature, a resolver requires the original TTL. [11] describes how to use the Original TTL field value to reconstruct the original TTL.

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

[3.1.5](#) Signature Expiration and Inception Fields

The Signature Expiration and Inception fields specify a validity period for the signature. The SIG record MUST NOT be used for authentication prior to the inception date and MUST NOT be used for authentication after the expiration date.

Signature Expiration and Inception field values are in POSIX.1 time format, a 32-bit unsigned number of seconds elapsed since 1 January 1970 00:00:00 UTC, ignoring leap seconds, in network byte order. The longest interval which can be expressed by this format without wrapping is approximately 136 years. A SIG RR can have an Expiration field value which is numerically smaller than the Inception field value if the expiration field value is near the 32-bit wrap-around point or if the signature is long lived. Because of this, all comparisons involving these fields MUST use "Serial number

arithmetic" as defined in [4]. As a direct consequence, the values contained in these fields cannot refer to dates more than 68 years in either the past or the future.

[3.1.6](#) The Key Tag Field

The Key Tag field contains the key tag value of the KEY RR that validates this signature. The process of calculating the Key Tag value is given in [Appendix B](#).

[3.1.7](#) The Signer's Name Field

The Signer's Name field value identifies the owner name of the KEY RR used to authenticate this signature. The Signer's Name field MUST contain the name of the zone of the covered RRset, unless the Type Covered field value is 0. A sender MUST NOT use DNS name compression on the Signer's Name field when transmitting a SIG RR. A receiver which receives a SIG RR containing a compressed Signer's Name field SHOULD decompress the field value.

Arends, et al.

Expires August 26, 2003

[Page 11]

Internet-Draft

DNSSEC Resource Records

February 2003

[3.1.8](#) The Signature Field

The Signature field contains the cryptographic signature which covers the SIG RDATA (excluding the Signature field) and the RRset specified by the SIG owner name, SIG class, and SIG Type Covered field.

[3.1.8.1](#) Signature Calculation

A signature covers the SIG RDATA (excluding the Signature Field) and covers the RRset specified by the SIG owner name, SIG class, and SIG Type Covered field. The RRset is in canonical form (see [Section 6](#)) and the set RR(1),...RR(n) is signed as follows:

signature = sign(SIG_RDATA | RR(1) | RR(2)...) where

"|" denotes concatenation;

SIG_RDATA is the wire format of the SIG RDATA fields with the Signer's Name field in canonical form and the Signature field excluded;

RR(i) = owner | class | type | TTL | RDATA length | RDATA;

"owner" is the fully qualified owner name of the RRset in canonical form (for RRs with wildcard owner names, the wildcard label is included in the owner name);

Each RR MUST have the same owner name as the SIG RR;

Each RR MUST have the same class as the SIG RR;

Each RR in the RRset MUST have the RR type listed in the SIG RR's Type Covered field;

Each RR in the RRset MUST have the TTL listed in the SIG Original TTL Field;

Any DNS names in the RDATA field of each RR MUST be in canonical form; and

The RRset MUST be sorted in canonical order.

[3.2](#) The SIG RR Presentation Format

The presentation format of the RDATA portion is as follows:

The Type Covered field value is represented either as an unsigned

decimal integer or as the mnemonic for the covered RR type.

The Algorithm field value is represented either as an unsigned decimal integer or as an algorithm mnemonic as specified in [Appendix A.1](#).

The Labels field value is represented as an unsigned decimal integer.

The Original TTL field value is represented as an unsigned decimal integer.

The Signature Inception Time and Expiration Time field values are represented in the form YYYYMMDDHHmmSS in UTC, where:

YYYY is the year (0000-9999, but see [Section 3.1.5](#));

MM is the month number (01-12);

DD is the day of the month (01-31);

HH is the hour in 24 hours notation (00-23);

mm is the minute (00-59);

SS is the second (00-59).

The Key Tag field is represented as an unsigned decimal integer.

The Signer's Name field value is represented as a fully qualified domain name.

The Signature field is represented as a Base64 encoding of the signature. Whitespace is allowed within the Base64 text. For a definition of Base64 encoding see [3] [Section 5.2](#).

[3.3](#) SIG RR Example

The following a SIG RR stores the signature for the A RRset of host.example.com:

```
host.example.com. 86400 IN SIG A 5 3 86400 20030322173103 (
    20030220173103 2642 example.com.
    oJB1W6WNGv+ldvQ3WDG0MQkg5IEhjRip8WTr
    PYGv07h108dUKGMeDPKijVCHX3DDKdfb+v6o
    B9wfuh3DTJXUAFI/M0zm0/zz8bW0Rznl803t
    GNazPwQKkRN20XPXV6nwwfoXmJQbsLNRlfkG
    J5D6fwFm8nN+6pBzeDQfsS3Ap3o= )
```

The first four fields specify the owner name, TTL, Class, and RR type (SIG). The "A" represents the Type Covered field. The value 5 identifies the Algorithm used (RSA-SHA1) to create the signature. The value 3 is the number of Labels in the original owner name. The value 86400 in the SIG RDATA is the Original TTL for the covered A RRset. 20030322173103 and 20030220173103 are the expiration and inception dates, respectively. 2642 is the Key Tag, and example.com. is the Signer's Name. The remaining text is a Base64 encoding of the

signature.

Note that combination of SIG RR owner name, class, and Type Covered indicate that this SIG covers the "host.example.com" A RRset. The Label value of 3 indicates that no wildcard expansion was used. The Algorithm, Signer's Name, and Key Tag indicate this signature can be authenticated using an example.com zone KEY RR whose algorithm is 5 and key tag is 2642.

The NXT resource record lists two separate things: the owner name of the next authoritative RRset in the canonical ordering of the zone, and the set of RR types present at the NXT RR's owner name. The complete set of NXT RRs in a zone both indicate which authoritative RRsets exist in a zone and also form a chain of authoritative owner names in the zone. This information is used to provide authenticated denial of existence for DNS data, as described in [11].

The type value for the NXT RR is 30.

The NXT RR is class independent.

[4.1](#) NXT RDATA Wire Format

The RDATA of the NXT RR is as shown below:

```

                                1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               Next Domain Name                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
/                               Type Bit Map                               /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

[4.1.1](#) The Next Domain Name Field

The Next Domain Name field contains the owner name of the next authoritative RRset in the canonical ordering of the zone; see [Section 6.1](#) for an explanation of canonical ordering. The value of the Next Domain Name field in the last NXT record in the zone is the name of the zone apex (the owner name name of the zone's SOA RR).

A sender MUST NOT use DNS name compression on the Next Domain Name field when transmitting an NXT RR. A receiver which receives an NXT RR containing a compressed Next Domain Name field SHOULD decompress the field value.

Owner names of non-authoritative RRsets (such as glue records) MUST NOT be listed in the Next Domain Name unless at least one authoritative RRset exists at the same owner name.

[4.1.2](#) The Type Bit Map Field

The Type Bit Map field identifies the RRset types which exist at the NXT RR's owner name.

Each bit in the Type Bit Map field corresponds to an RR type. Bit 1 corresponds to RR type 1 (A), bit 2 corresponds to RR type 2 (NS), and so forth. If a bit is set to 1, it indicates that an RRset of that type is present for the NXT's owner name. If a bit is set to 0, it indicates that no RRset of that type present for the NXT's owner name.

Bit 1 MUST NOT indicate glue address records.

Bit 41 MUST have the value of 0, since the OPT pseudo-RR [6] can never appear in zone data.

Trailing zero octets MUST be omitted. The length of the Type Bit Map field varies, and is determined by the type code with the largest numerical value among the set of RR types present at the NXT RR's owner name. Trailing zero octets not specified MUST be interpreted as zero octets.

The above Type Bit Map format MUST NOT be used when an RR type code with numerical value greater than 127 is present.

Bit 0 in the Type Bit Map field indicates the Type Bit Map format. A value of 0 in bit 0 denotes the format described above, therefore bit 0 MUST have a value of 0. The format and meaning of a Type Bit Map with a value of 1 in bit 0 is undefined.

[4.1.3](#) Inclusion of Wildcard Names in NXT RDATA

If a wildcard owner name appears in a zone, the wildcard label ("*") is treated as a literal symbol and is treated the same as any other owner name for purposes of generating NXT RRs. Wildcard owner names appear in the Next Domain Name field without any wildcard expansion. [11] describes the impact of wildcards on authenticated denial of existence.

[4.2](#) The NXT RR Presentation Format

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 either as a sequence of RR type mnemonics or as a sequence of unsigned decimal integers denoting the RR type codes.

[4.3](#) NXT RR Example

The following NXT RR identifies the RRsets associated with

Arends, et al.

Expires August 26, 2003

[Page 16]

Internet-Draft

DNSSEC Resource Records

February 2003

alfa.example.com. and identifies the next authoritative name after
alfa.example.com.

alfa.example.com. 86400 IN NXT host.example.com. A MX SIG NXT

The first four text fields specify the name, TTL, Class, and RR type (NXT). The entry host.example.com. is the next authoritative name after alfa.example.com. (in canonical order). The A, MX, SIG and NXT mnemonics indicate there are A, MX, SIG and NXT RRsets associated with the name alfa.example.com.

Note the NXT record can be used for authenticated denial of existence. If the example NXT record were authenticated, it could be used to prove that beta.example.com. does not exist, or could be used to prove there is no AAAA record associated with alfa.example.com. Authenticated denial of existence is discussed in [\[11\]](#)

[5.](#) The DS Resource Record

The DS Resource Record refers to a KEY RR and is used in the DNS KEY authentication process. A DS RR refers to a KEY RR by storing the key tag, algorithm number, and a digest of KEY RR. Note that while the digest should be sufficient to identify the key, storing the key tag and key algorithm helps make the identification process more efficient. By authenticating the DS record, a resolver can authenticate the KEY RR to which the DS record points. The key authentication process is described in [\[11\]](#).

The DS RR and its corresponding KEY RR have the same owner name, but they are stored in different locations. The DS RR appears only on the upper (parental) side of a delegation, and is authoritative data in the parent zone. For example, the DS RR for "example.com" is stored in the "com" zone (the parent zone) rather than in the "example.com" zone (the child zone). The corresponding KEY RR is stored in the "example.com" zone (the child zone). This simplifies DNS zone management and zone signing, but introduces special response processing requirements for the DS RR; these are described in [\[11\]](#).

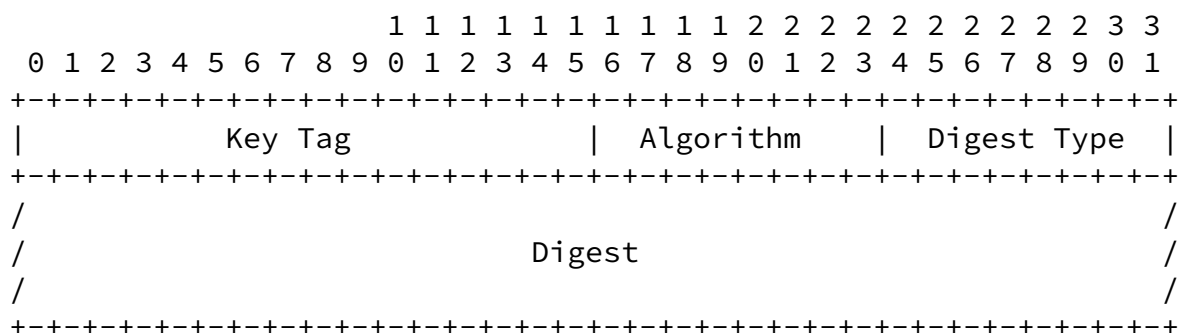
The type number for the DS record is 43.

The DS resource record is class independent.

There are no special TTL requirements on the DS resource record.

[5.1](#) DS RDATA Wire Format

The RDATA for a DS RR consists of 2 octet Key Tag field, a one octet Algorithm field, a one octet Digest Type field, and a Digest field.



5.1.1 The Key Tag Field

The Key Tag field lists the key tag of the KEY RR referred to by the DS record. The KEY RR MUST be a zone key. The KEY RR Flags MUST have Flags bit 7 set to value 1.

The Key Tag used by the DS RR is identical to the Key Tag used by the SIG RR and [Appendix B](#) describes how to compute a Key Tag.

5.1.2 The Algorithm Field

The Algorithm field lists the algorithm number of the KEY RR referred to by the DS record.

The algorithm number used by the DS RR is identical to the algorithm number used by the SIG RR and KEY RR. [Appendix A.1](#) lists the algorithm number types.

5.1.3 The Digest Type Field

The DS RR refers to a KEY RR by including a digest of that KEY RR. The Digest Type field identifies the algorithm used to construct the digest and [Appendix A.2](#) lists the possible digest algorithm types.

5.1.4 The Digest Field

The DS record refers to a KEY RR by including a digest of that KEY RR. The Digest field holds the digest.

The digest is calculated by concatenating the canonical form of the fully qualified owner name of the KEY RR (abbreviated below as "key

RR name") with the KEY RDATA, and then applying the digest algorithm.

```
digest = digest_algorithm( KEY RR name | KEY RDATA);
```

"|" denotes concatenation

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

The size of the digest may vary depending on the digest algorithm and KEY RR size. Currently, the defined digest algorithm is SHA-1, which produces a 20 octet digest.

[5.2](#) The DS RR Presentation Format

The presentation format of the RDATA portion is as follows:

The Key Tag field is represented as an unsigned decimal integer.

The Algorithm field is represented either as an unsigned decimal integer or as an algorithm mnemonic specified in [Appendix A.1](#).

The Digest Type field is represented as an unsigned decimal integer.

The Digest is represented as a sequence of case-insensitive hexadecimal digits. Whitespace is allowed within the hexadecimal text.

[5.3](#) DS RR Example

The following example shows a KEY RR and its corresponding DS RR.

```
dskey.example.com. 86400 IN KEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz9Xz
fwJr1AYtsmx3TGkJaNXVbfi/
2pHm822aJ5iI9BMzNXxeYcmZ
DRD99WYwYqUSdjMmmAphXdvx
egXd/M5+X70rzKBaMbCVdFLU
Uh6DhweJBjEVv5f2wwjM9Xzc
nOf+EPbtG9DMbMADjFDc2w/r
ljwvFw==
) ; key id = 60485
```

```
dskey.example.com. 86400 IN DS 60485 5 1 ( 2BB183AF5F22588179A53B0A
                                           98631FAD1A292118 )
```

The first four text fields specify the name, TTL, Class, and RR type (DS). Value 60485 is the key tag for the corresponding "dskey.example.com." KEY RR, and value 5 denotes the algorithm used by this "dskey.example.com." KEY RR. The value 1 is the algorithm used to construct the digest, and the rest of the RDATA text is the digest in hexadecimal.

[6.](#) Canonical Form and Order of Resource Records

This section defines a canonical form for resource records, a canonical ordering of DNS names, and a canonical ordering of resource records within an RRset. A canonical name order is required to construct the NXT name chain. A canonical RR form and ordering within an RRset are required to construct and verify SIG RRs.

[6.1](#) Canonical DNS Name Order

For purposes of DNS security, owner names are ordered by treating

individual labels as unsigned left-justified octet strings. The absence of a octet sorts before a zero value octet, and upper case US-ASCII letters are treated as if they were lower case US-ASCII letters.

To compute the canonical ordering of a set of DNS names, start by sorting the names according to their most significant (rightmost) labels. For names in which the most significant label is identical, continue sorting according to their next most significant label, and so forth.

For example, the following names are sorted in canonical DNS name order. The most significant label is "example". At this level, "example" sorts first, followed by names ending in "a.example", then names ending "z.example". The names within each level are sorted in the same way.

```
example
a.example
ylkjlk.a.example
Z.a.example
zABC.a.EXAMPLE
z.example
\001.z.example
*.z.example
\200.z.example
```

[6.2](#) Canonical RR Form

For purposes of DNS security, the canonical form of an RR is the wire format of the RR where:

1. Every domain name in the RR is fully expanded (no DNS name compression) and fully qualified;
2. All uppercase US-ASCII letters in the owner name of the RR are

replaced by the corresponding lowercase US-ASCII letters;

3. If the type of the RR is NS, MD, MF, CNAME, SOA, MB, MG, MR, PTR, HINFO, MINFO, MX, HINFO, RP, AFSDB, RT, SIG, PX, NXT, NAPTR, KX,

SRV, DNAME, or A6, all uppercase US-ASCII letters in the DNS names within the RDATA of the RR are replaced by the corresponding lowercase US-ASCII letters;

4. If the owner name of the RR is a wildcard name, the owner name is in its original unexpanded form, including the "*" label (no wildcard substitution); and
5. The RR's TTL is set to its original value as it appears in the authoritative zone containing the RR or the Original TTL field of the covering SIG RR.

Editors' Note: the above definition sacrifices readability for an attempt at precision. Please send better text!

[6.3](#) Canonical RR Ordering Within An RRset

For purposes of DNS security, RRs with same owner name, same class, and same type are sorted by sorting the canonical forms of the RRs while treating the RDATA portion of the canonical form of each RR as a left justified unsigned octet sequence. The absence of an octet sorts before the zero octet.

7. IANA Considerations

This document introduces one new IANA consideration. [RFC 2535](#) [14] created an IANA registry for DNS Security Algorithm Numbers. This document re-assigns DNS Security Algorithm Number 252 to be "reserved". This value is no longer available for assignment by IANA.

This document clarifies the use of existing DNS resource records. For completeness, the IANA considerations from the previous documents which defined these resource records are summarized below. No IANA changes are made by this document other than the one change described in the first paragraph of this section.

[14] updated the IANA registry for DNS Resource Record Types, and assigned types 24,25, and 30 to the SIG, KEY, and NXT RRs, respectively. [9] assigned DNS Resource Record Type 43 to DS.

[14] created an IANA registry for DNSSEC Resource Record Algorithm Numbers. Values to 1-4, and 252-255 were assigned by [14]. Value 5 was assigned by [8]. Value 252 is re-assigned by this document, as noted above.

[9] created an IANA registry for DNSSEC DS Digest Types, and assigned value 0 to reserved and value 1 to SHA-1.

[14] created an IANA Registry for KEY Protocol Values, but [16] re-assigned all assigned values other than 3 to reserved and closed this IANA registry. The registry remains closed, and all KEY records are required to have Protocol Octet value of 3.

The Flag bits in the KEY RR are not assigned by IANA, and there is no IANA registry for these flags. All changes to the meaning of the KEY RR Flag bits require a standards action.

The meaning of a value of 1 in bit zero of the Type Bit Map of an NXT RR can only be assigned by a standards action.

[8](#). Security Considerations

This document describes the format of four DNS resource records used by the DNS security extensions, and presents an algorithm for calculating a key tag for a public key. Other than the items described below, the resource records themselves introduce no security considerations. The use of these records is specified in a separate document, and security considerations related to the use these resource records are discussed in that document.

The DS record points to a KEY RR using a cryptographic digest, the key algorithm type and a key tag. The DS record is intended to identify an existing KEY RR, but it is theoretically possible for an attacker to generate a KEY that matches all the DS fields. The probability of constructing such a matching KEY depends on the type of digest algorithm in use. The only currently defined digest algorithm is SHA-1, and the working group believes that constructing a public key which would match the algorithm, key tag, and SHA-1 digest given in a DS record would be a sufficiently difficult problem that such an attack is not a serious threat at this time.

The key tag is used to help select KEY resource records efficiently, but it does not uniquely identify a single KEY resource record. It is possible for two distinct KEY RRs to have the same owner name, the same algorithm type, and the same key tag. An implementation which used only the key tag to select a KEY RR might select the wrong public key in some circumstances. Implementations MUST NOT assume the key tag is unique public key identifier; this is clearly stated in [Appendix B](#).

[9](#). Acknowledgments

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 revision of these security extension specifications.

Normative 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] Borenstein, N. and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", [RFC 1521](#), September 1993.
- [4] Elz, R. and R. Bush, "Serial Number Arithmetic", [RFC 1982](#), August 1996.
- [5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [6] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", [RFC 2671](#), August 1999.
- [7] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", [RFC 2931](#), September 2000.
- [8] Eastlake, D., "RSA/SHA-1 SIGs and RSA KEYs in the Domain Name

System (DNS)", [RFC 3110](#), May 2001.

- [9] Gudmundsson, O., "Delegation Signer Resource Record", [draft-ietf-dnsext-delegation-signer-12](#) (work in progress), December 2002.
- [10] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "DNS Security Introduction and Requirements", [draft-ietf-dnsext-dnssec-intro-05](#) (work in progress), February 2003.
- [11] Arends, R., Austein, R., Larson, M., Massey, D. and S. Rose, "Protocol Modifications for the DNS Security Extensions", [draft-ietf-dnsext-dnssec-protocol-00](#) (work in progress), Februari 2003.
- [12] Gustafsson, A., "Handling of Unknown DNS RR Types", [draft-ietf-dnsext-unknown-rrs-04](#) (work in progress), September 2002.
- [13] Kisters, M., Blacka, D. and R. Arends, "DNSSEC Opt-in", [draft-ietf-dnsext-dnssec-opt-in-04](#) (work in progress), February 2003.

Arends, et al.

Expires August 26, 2003

[Page 26]

Internet-Draft

DNSSEC Resource Records

February 2003

Informative References

- [14] Eastlake, D., "Domain Name System Security Extensions", [RFC 2535](#), March 1999.
- [15] Eastlake, D., "Secret Key Establishment for DNS (TKEY RR)", [RFC 2930](#), September 2000.
- [16] Massey, D. and S. Rose, "Limiting the Scope of the KEY Resource Record (RR)", [RFC 3445](#), December 2002.

Authors' Addresses

Roy Arends
Telematica Instituut
Drienerlolaan 5
7522 NB Enschede
NL

E-Mail: roy.arends@telin.nl

Rob Austein
Internet Software Consortium
40 Gavin Circle
Reading, MA 01867
USA

E-Mail: sra@isc.org

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

E-Mail: mlarson@verisign.com

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

E-Mail: masseyd@isi.edu

Arends, et al.

Expires August 26, 2003

[Page 27]

Internet-Draft

DNSSEC Resource Records

February 2003

Scott Rose
National Institute for Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-8920
USA

E-Mail: scott.rose@nist.gov

[Appendix A](#). DNSSEC Algorithm and Digest Types

The DNS security extensions are designed to be independent of the underlying cryptographic algorithms. The KEY, SIG, and DS resource records all use a DNSSEC Algorithm Number to identify the

cryptographic algorithm in use by the resource record. The DS resource record also specifies a Digest Algorithm Number to identify the digest algorithm used to construct the DS record. The currently defined Algorithm and Digest Types are listed below. Additional Algorithm or Digest Types could be added as advances in cryptography warrant.

A DNSSEC aware resolver or name server MUST implement all MANDATORY algorithms.

[A.1](#) DNSSEC Algorithm Types

An "Algorithm Number" field in the KEY, SIG, and DS resource record types identifies the cryptographic algorithm used by the resource record. Algorithm specific formats are described in separate documents. The following table lists the currently defined algorithm types and provides references to their supporting documents:

VALUE	Algorithm	RFC	STATUS
0	Reserved	-	-
1	RSA/MD5	RFC 2537	NOT RECOMMENDED
2	Diffie-Hellman	RFC 2539	OPTIONAL
3	DSA	RFC 2536	OPTIONAL
4	elliptic curve	TBA	OPTIONAL
5	RSA/SHA1	RFC 3110	MANDATORY
6-251	available for assignment	-	
252	reserved	-	
253	private	see below	OPTIONAL
254	private	see below	OPTIONAL
255	reserved	-	-

[A.1.1](#) Private Algorithm Types

Algorithm number 253 is reserved for private use and will never be assigned to a specific algorithm. The public key area in the KEY RR and the signature area in the SIG RR begin with a wire encoded domain name. Only local domain name compression is permitted. The domain name indicates the private algorithm to use and the remainder of the public key area is determined by that algorithm. Entities should only use domain names they control to designate their private algorithms.

Algorithm number 254 is reserved for private use and will never be assigned to a specific algorithm. The public key area in the KEY RR and the signature area in the SIG RR 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 OIDs they control to designate their private algorithms.

[A.2](#) DNSSEC Digest Types

A "Digest Type" field in the DS resource record types identifies the cryptographic digest algorithm used by the resource record. The following table lists the currently defined digest algorithm types.

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

[Appendix B](#). Key Tag Calculation

The Key Tag field in the SIG and DS resource record types provides a mechanism for selecting a public key efficiently. In most cases, a combination of owner name, algorithm, and key tag can efficiently identify a KEY record. Both the SIG and DS resource records have corresponding KEY records. The Key Tag field in the SIG and DS records can be used to help select the corresponding KEY RR efficiently when more than one candidate KEY RR is available.

However, it is essential to note that the key tag is not a unique identifier. It is theoretically possible for two distinct KEY RRs to have the same owner name, the same algorithm, and the same key tag. The key tag is used to limit the possible candidate keys, but it does not uniquely identify a KEY record. Implementations **MUST NOT** assume that the key tag uniquely identifies a KEY RR.

The key tag is the same for all KEY algorithm types except algorithm 1 (please see [Appendix B.1](#) for the definition of the key tag for algorithm 1). For all algorithms other than algorithm 1, the key tag is defined to be the output which would be generated by running the ANSI C function shown below with the RDATA portion of the KEY RR as input. It is not necessary to use the following reference code verbatim, but the numerical value of the Key Tag **MUST** be identical to what the reference implementation would generate for the same input.

Please note that the algorithm for calculating the Key Tag is almost but not completely identical to the familiar ones complement checksum used in many other Internet protocols. Key Tags **MUST** be calculated using the algorithm described below rather than the ones complement checksum.

The following ANSI C reference implementation calculates the value of a Key Tag. This reference implementation applies to all algorithm types except algorithm 1 (see [Appendix B.1](#)). The input is the wire format of the RDATA portion of the KEY RR. The code is written for clarity, not efficiency.

```
/*
 * Assumes that int is at least 16 bits.
 * First octet of the key tag is the most significant 8 bits of the
 * return value;
 * Second octet of the key tag is the least significant 8 bits of the
 * return value.
 */

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

    for ( ac = 0, i = 0; i < keysize; ++i )
        ac += (i & 1) ? key[i] : key[i] << 8;
    ac += (ac >> 16) & 0xFFFF;
    return ac & 0xFFFF;
}
```

[B.1](#) Key Tag for Algorithm 1 (RSA/MD5)

The key tag for algorithm 1 (RSA/MD5) is defined differently than the key tag for all other algorithms, for historical reasons. For a KEY RR with algorithm 1, the key tag is defined to be the most significant 16 bits of the least significant 24 bits in the public key modulus (in other words, the 4th to last and 3rd to last octets of the public key modulus).

Please note that Algorithm 1 is NOT RECOMMENDED.

Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

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