Network Working Group
Internet-Draft
Expires: April 29, 2003

R. Arends

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

October 29, 2002

Resource Records for the DNS Security Extensions
draft-ietf-dnsext-dnssec-records-02

Status of this Memo

Copyright Notice

Abstract

This document is part of a family of documents that describe 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 KEY, DS, SIG, and NXT resource records. The purpose and format of each resource record is descibed in detail and an example of each

resource record is given.

This document obsoletes RFC 2535 and incorporates changes from all updates to RFC 2535.

Table of Contents

1. Introduction

The DNS Security Extensions (DNSSEC) introduce four resource records: the KEY, SIG, NXT, and DS resource records. This document defines the purpose of each resource record (RR), the RR's RDATA format, and its ASCII representation. An example of each RR type is also given.

1.1 Background and Related Documents

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 [13]. A description of DNS protocol modifications can be found in [14]. This document defines the DNSSEC resource records.

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

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 [4].

1.3 Editors Notes

1.3.1 Open Technical Issues

The NXT section (Section 4) requires input from the working group. Since the opt-in issue is not resolved, this text describes the NXT record as it was defined in RFC 2535. This section may need to be updated, depending on the outcome of the opt-in discussion.

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 seperate internet draft solely to move DSA from MANDATORY to OPTIONAL seemed excessive. This draft solicts input on that proposed change.

The indirect and private algorithms types (Appendix A) are also worth noting. See the text in that section.

1.3.2 Technical Changes or Corrections

Please report technical corrections to dnssec-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 there is no RFC that defines the correct behavior (or RFCs provide conflicting answers), please post the issue to namedroppers.

An example correction to dnssec-editors 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 dnssec-editors@east.isi.edu. To assist the editors, please provide enough context for us to quickly find the incorrect text.

An example message to dnssec-editors 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 cryptogrpahy 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 [14]. In a typical example, a zone signs its authorititave 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 is also used to store public keys associated with other DNS operations, such as SIG(0) [14] and TKEY [9]. 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.

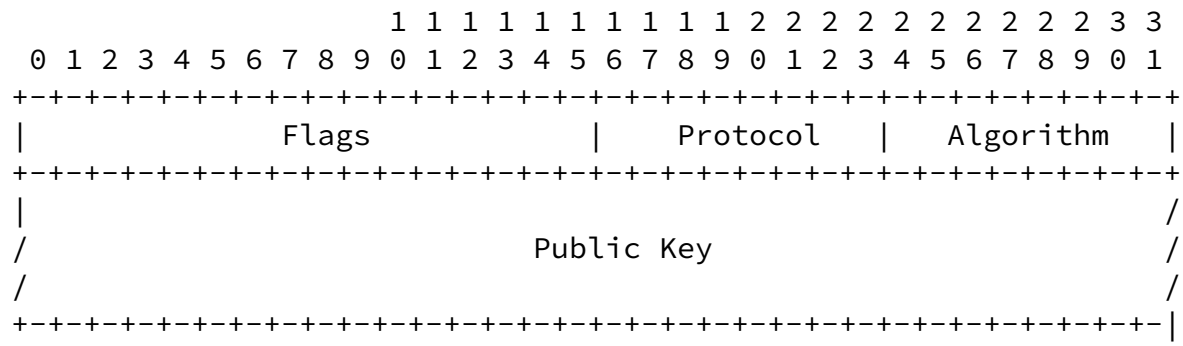
The type number for the KEY RR is 25.

The KEY RR is class independent.

There are no special TTL requirements on the KEY record. DNSSEC best practices documents are encouraged to provide TTL recommendations.

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. If bit 7 is 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 is 0, then the KEY record holds some

other type of DNS public key, such as a public key used by SIG(0) or TKEY.

Bits 0-6 and 8-15 are reserved for future use and MUST be zero.

2.1.2 The Protocol Octet Field

The Protocol Octet field MUST be 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. A list of DNSSEC algorithm types can be found in Appendix A.1

2.1.4 Notes on KEY RDATA Design

Although the Protocol Octet field is always 3, it is retained for backwards compatibility with an earlier version of the KEY record. The use of bit 7 as the Zone Key Flag is also due to backwards compatiblity issues.

2.2 The KEY RR Presentation Format

A KEY RR may appear as a single line or multiple lines separated with newline characters if those lines are contained with parantheses. 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 an algorithm mnemonic specified in Appendix A.1.

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

2.3 KEY RR Example

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==)
```

The first four fields specify the owner name, TTL, Class, and RR type (KEY). 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. 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 [12]. The remaining text is a base 64 encoding of the public key.

3. The SIG Resource Record

DNSSEC uses public key cryptogrpahy to sign and authenticate DNS resource record sets (RRsets). The signatures are stored in SIG resource records and are used in the DNSSEC authentication process described in [14]. In a typical example, a zone signs its authorititave RRsets using a private key and stores the corresponding signatures in SIG RRs. A resolver can then use these signatures 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 is said to "cover" this RRset. The SIG RR also specifies a validity interval for the signature and uses an algorithm signer's name, and key tag to identify the public key (KEY record) that can be used to verify the signature.

The signature in SIG RR may also cover a transaction rather than an RRset [14]. In this case, the "Type Covered" field is set to 0 and the SIG RR is refered to as SIG(0) resource record.

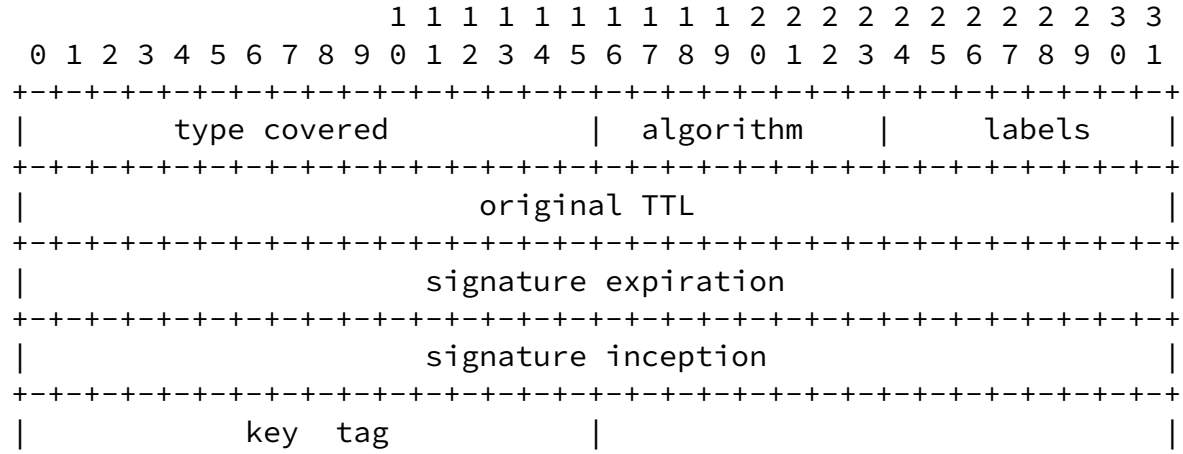
The type number for the SIG RR type is 24.

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

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

3.1 The SIG RDATA

The RDATA portion of a SIG RR is shown below:



3.1.1 The Type Covered Field

The Type Covered field identifies the RRset type covered by the SIG record.

If Type Covered field is set to 0, the record is referred to as a SIG(0) RR and its signature covers a transaction rather than a specific RRset. [14] descirbes how to sign transactions using SIG(0) resource records.

3.1.2 The Algorithm Number Field

The Algorithm Number field identies 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 the signature, a resolver requires the original owner name that was used when the signature was created. In most cases, the owner name used when the signature was created is identical to the owner name sent in any response. However, a wildcard owner name will be expanded during the query/response process and [14] describes how the label count is used to reconstruct the original (unexpanded) owner name.

The Labels field does not count null labels for root and does not count any initial "*" in a wildcard name. The Labels 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 count of 3 and "*.example.com." has a label count of 2.

3.1.4 Original TTL Field

The Original TTL field specifies the original TTL of the covered RRset.

To validate the signature, a resolver requires the original TTL used when the signature was created. However, caching servers will decrement the TTL and [14] describes how the Original TTL field count

is used to reconstruct the original (undecremented) TTL.

If the Type Covered field is non-zero, the Original TTL value MUST be greater than or equal to the TTL of the SIG record itself. If the Type Covered field is 0 (i.e. a SIG(0) RR), the Original TTL field SHOULD be zero.

3.1.5 Signature Expiration and Inception Fields

The Signature Inception and Signature Expiration 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 expiratiation date.

Inception and expiration dates are given as 32-bit unsigned numbers of seconds since the start of 1 January 1970 GMT, ignoring leap seconds. Ring arithmetic [3] to handle 32-bit wrap around. As result, these times can never be more than 68 years in the past or the future and the times are ambiguous modulo ~136 years. A SIG RR can have an expiration time numerically smaller 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 field contains the key tag of the public key (KEY RR) used to authenticate this signature. The process of calculating a key tag is given in Appendix B.

3.1.7 The Signer's Name Field

The Signer's Name field identifies the name of the KEY RR used to authenticate this signature. If the Type Covered field is non-zero, the Signer's Name MUST contain the name of the zone containing the covered RRset and the SIG. The signer's name MAY be compressed with standard DNS name compression when being transmitted over the network.

If the Type Covered field is 0 (i.e. a SIG(0) RR), the signer's name MUST be the name of the host originating the DNS message as described in [10].

3.1.8 The Signature Field

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

3.2 Calculating A Signature

A signature covers either an RRset or a transaction. RRset signatures and transaction signatures are distinguished by the Type Covered field. RRset signatures have a non-zero Type Covered field. SIG RRs SHOULD NOT be generated for any "meta-type" such as ANY or AXFR.

3.2.1 Calculating An RRset Signature

A signature covers the SIG RDATA (excluding the Signature Field itself) and covers the RRset specified by the SIG owner name, SIG class, and SIG Type Covered field. The RRset is in cannonical form (see Appendix C) and the set RR(1),...RR(n) is signed as follows:

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

"|" denotes append

SIG_RDATA is the wire format of the SIG RDATA fields with the Signer's Name field in cannonical form. the Signature field excluded.

RR(i) = fqdn | class | type | TTL | RDATA length | RDATA

fqdn is the Fully Qualified Domain Name in canonical form.

All RR(i) MUST have the same fqdn as the SIG RR.

All RR(i) MUST have the same class as the SIG RR.

All RR(i) MUST have the RR type listed in SIG RR's Type Covered field.

All RR(i) MUST have the TTL listed in the SIG Original TTL Field

All names in the RDATA field are in canonical form

The set of all RR(i) is sorted into cannonical order.

3.2.2 Calculating An Transaction Signature

3.3 The SIG RR Presentation Format

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

The Type Covered field is represented by either an unsigned integer or the mnemonic for the RR type.

The Algorithm field is represented as an unsigned integer or as an algorithm mnemonic specified in Appendix A.1.

The Labels field is represented as an unsigned integer.

The Original TTL field is represented as an unsigned integer. It MAY be omitted if it is equal the TTL of the SIG RR.

The Signature Inception Time and Expiration Time fields are represented in the form YYYYMMDDHHmmSS, where:

YYYY is the year

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 integer.

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

The Signature field is a Base 64 encoding of the signature.

3.4 Example of a SIG RR

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

```
host.example.com.  30 IN     SIG     A 3 3 30 20011231120000 (
                                     20011108100000 65531 example.com
                                     CGr0uS55C4l/2RRc2NrMJbRt4oP+xVxwgMkC
                                     rJFXXDsybfEDdwoajAY= )
```

The first four fields specify the owner name, TTL, Class, and RR type

(SIG). The "A" represents the Type Covered field. is the algorithm used to create this signature. The first 3 identifies the Algorithm used to create the signature. The second 3 is the number of Labels in the original owner name and the 30 is the Original TTL for this SIG RR and the covered A RRset. The two dates are the expiration and inception dates. 65531 is the Key Tag and example.com. is the Signer's Name. The remaining text is a base 64 encoding of the signature.

Note that combination of SIG RR owner name, class, and and Type Covered indicate this SIG covers the "host.example.com" A RRset. The Label value of 3 indicates 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 3 and key tag is 65531.

4. The NXT Resource Record

The NXT resource record lists the RR types present at the NXT's owner name and lists the next canonical name in the zone. The collection of NXT or "next" resource records indicate what RRsets exist in a zone and provide a chain of all authoritative owner names in that zone. This information can be used for authenticated denial of existence, as desribed in [14].

Note that although a zone may contain non-authoritiative glue address records, these non-authoritative glue records MUST NOT be used when contructing the NXT resource record chain.

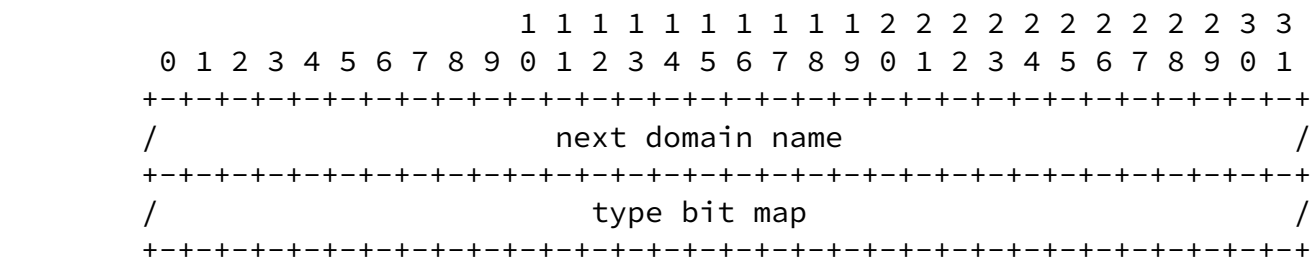
The type number for the NXT RR is 30.

The NXT RR is class independent.

The NXT RR TTL SHOULD NOT exceed the minimum TTL in the zone's SOA RR.

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 authoritive owner name in canonical order, where canonical order is defined in Appendix C.1. For the last owner name in the zone, the Next Domain Name field contains the zone apex name.

The Next Domain Name field allows message compression.

Note that non-authoritative glue address record names may exist in a zone, but these non-authoritative glue records MUST NOT be listed in the Next Domain Name. Any non-authoritative glue records are ignored (treated as though they were never present) when constructing an NXT.

4.1.2 The Type Bit Map Field

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

Each bit in the Type Bit Map field corresponds to an RR type. Bit one 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 exists for the NXT's owner name. If a bit is set to zero, it indicates that no RRset of that type exists for the NXT's owner name.

Trailing zero octets MUST be omitted. Thus the length of the Type Bit Map field varies and is dependent on the largest RR type present for the NXT's owner name. Trailing zero octets not specified MUST be interpreted as zero octets.

Non-authoritative glue address record types MUST NOT be used when constructing the type bit map field. The OPT RR [8] type (41) also MUST NOT be used when constructing the type bit map field since it is not part of the zone data. In other words, the OPT RR type bit (bit 41) MUST be zero.

4.1.2.1 Alternate Formats for the Type Bit Map Field

The above Type Bit Map format MUST NOT be used when an RR type number greater than 127 is in use.

Bit 0 in the Type Bit Map Field is used to indicate an alternate format for the Type Bit Map field. If bit 0 is set to 1, it indicates some other format is being used for this field. No alternate formats are defined as of this writing.

4.1.3 Inclusion of Wildcard Names in NXT RDATA

If a wildcard owner name appears in a zone, the wildcard is treated as a literal symbol and is treated the same as any other owner name. Wildcard owner names appear (unexpanded) in the Next Domain Name field without any wildcard expansion. [14] describes the impact of wildcards on authetnicated denial of existence.

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 a sequence of unsigned integers denoting the RR types.

4.3 NXT RR Example

The following NXT RR identifies the RRsets associated with a.example.com and identifies the next authoritative name after a.example.com.

```
a.example.com. 86400 IN NXT c.example.com. A MX NXT
```

The first four fields specify the name, TTL, Class, and RR type (NXT). The entry c.example.com is the next authoritative name after a.example.com (in cannonical order). The A MX and NXT nnemonics indicate there are A, MX, and NXT RRsets associated with the name a.example.com.

Note the NXT record can be used for authenticted denial of existence. If the example NXT record were authenticed, it could be used to prove that b.example.com does not exist or could be used to prove there is no AAAA record assoicated with a.example.com. Authenticated denial of existence is discussed in [14]

5. The DS Resource Record

The DS Resource Record points to a KEY RR and is used in the DNS KEY authentication process. A DS RR points 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 key, storing the key tag and key algorithm helps make the identification process more efficient and more secure. By authenticating the DS record, a resolver can authenticate the KEY RR pointed to by the DS record. The key authentication proces is described in [14].

The DS RR and its corresponding KEY RR both have the same owner name, but they are stored in different locations. The DS RR is the first resource record that appears only on the upper side of a delegation. In other words, the DS RR for "example.com" is stored in "com" (the upper side of the delegation). The corresponding KEY RR is stored in the "example.com" zone (the lower side of the delegation). This simplifies DNS zone management and zone signing, but introduces special response processing requirements that are described in [14].

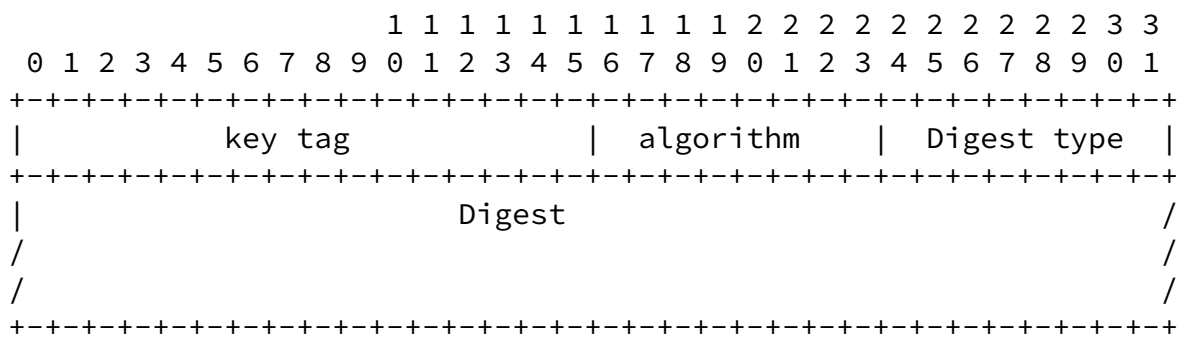
The type number for the DS record is 43.

The DS resource record is class independent.

There are no special TTL requirements on the DS resource record. DNSSEC best practices documents are encouraged to provide TTL recommendations.

5.1 DS RDATA Wire Format

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



5.1.1 The Key Tag Field

The Key Tag field lists the key tag of the KEY RR pointed to by the DS record. The KEY RR MUST be a a zone key. In other words, the KEY

RR Flags must have Flags bit 7 set to 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 pointed 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 points to a KEY RR by including a digest of that KEY RR. The Digest Type field identifes 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 points to a KEY RR by including a digest of that KEY RR. The Digest field hold the digest.

For a given KEY RR, the digest is calculated by appending the KEY RR's cannonical fully qualified owner name with the KEY RDATA and then applying the digest algorithm.

```
digest = digest_algorithm( cannonical FQDN of KEY RR | KEY_RR_rdata)

   "|" denotes append

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

The size of the digest can vary depending on the digest algorithm and KEY RR size. However, the only currently defined digest algorithm is SHA-1 and it always produces a 24 byte digest regardless of KEY RR size.

5.2 The DS RR Presentation Format

A DS RR may appear as a single line or multiple lines separated with newline characters if those lines are contained within parantheses. The presentation format of the RDATA portion is as follows:

The Key Tag field is represented as an unsigned integer.

The Algorithm field is represented as an unsigned integer or as an algorithm mnemonic specified in Appendix A.1.

The Digest Type field is represented as an unsigned integer.

The Digest is presented in hexadecimal.

5.3 DS Record Example

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

```
dskey.example. 86400 IN KEY  256 3 1 ( AQPwHb4UL1U9RHaU8qP+Ts5bVOU
                                       1s7fYbj2b3CCbzNdj4+/ECd18yKiy
                                       UQqKqQFWW5T3iVc8SJOKnueJHt/Jb
                                       /wt) ; key tag = 28668
dskey.example. 3600 IN  DS   28668 1  1  49FD46E6C4B45C55D4AC69CBD3CD34AC1AF
```

The first four fields specify the name, TTL, Class, and RR type (DS).

28668 is the key tag for the corresponding "dskey.example." KEY RR and 1 algorithm used by this "dskey.example." KEY RR. The second 1 is the algorithm used to construct the digest and the final string is the digest in hex.

6. IANA Considerations

This document introduces no new IANA considerations.

This document only clarifies the use of existing DNS resource records. However for completeness, the IANA considerations from these previous documents are summarized below. No IANA changes are made by this document.

RFC 2535 updated the IANA registry for DNS Resource Record Types and assigned types 24,25, and 30 to the SIG, KEY, and NXT (respectively).

[DS RFC] assigned DNS Resource Record Type 43 to DS.

RFC 2535 created an IANA registry for DNSSEC Resource Record Algorithm Numbers. Values to 1-4, and 252-255 were assigned by RFC 2535. Value 5 was assigned by RFC 3110.

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

RFC 2535 created an IANA Registry to KEY Protocol Octet Values, but [KeyRestrict RFC] set 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.

7. 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 desribed below, the resource records themselves introduce no security considerations. The use of these records is specified in a seperate 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 possibile 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 and the only currently defined digest algorithm is SHA1. It is considered very difficult to constuct a public key that matches the algorithm, key tag, and SHA1 digest given in a DS record.

The key tag is used to help efficiently select KEY resource records, but it does not uniquely identify a KEY resource record. It is possible that two distinct KEY RRs could have the same owner name, same algorithm type and same key tag. An implementation that used only the key tag to select a KEY RR may select the wrong public key for a given scenario. Implementations MUST NOT assume the key tag is unique public key identifier and this is clearly stated in the text.

8. 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 revision 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", October 2002.

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

[15] Massey, D. and S. Rose, "Limiting the Scope of the KEY Resource Record", draft-ietf-dnsext-restrict-key-for-dnssec-02 (work in progress), March 2002.

Authors' Addresses

Roy Arends
Bankastraat 41-E
1094 EB Amsterdam
NL

EMail: roy@logmess.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 20899-8920
USA

EMail: scott.rose@nist.gov

Appendix A. DNSSEC Algorithm and Digest Types

The DNS security exstentions 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 crytographic 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	indirect	see below	OPTIONAL
253	private	see below	OPTIONAL
254	private	see below	OPTIONAL
255	reserved	-	-

A.1.1 Indiret and Private Algorithm Types

RFC 2535 describes Algorithm number 252 as an indirect key format where the actual key material is elsewhere. This format was to be defined in a separate document. In the years between RFC 2535 and this document, no indirect key document has been prodcued.

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.

Editors Note: There is currently no use of or operational experience with these algorithms. The editors (at least Dan!) recommend that these algorithm types be eliminated. We don't need this in the base spec and every other algorithm type requires a seperate document to describe it in detail. Note eliminating these from the base spec

would not elminate any future functionality since there are 200+ available algorithm numbers. Anyone who feels they need this type of algorithm (or a similar algorithm) can write the document clearly describing it.

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	RSA/SHA-1	MANDATORY
2-255	Unassigned	-

Appendix B. Key Tag Calculation

The key tag field provides a mechanism for efficiently selecting a public key. In most cases, a combination of owner name, algorithm, and key tag can efficiently identify a KEY record. For example, both the SIG and DS resource records have corresponding KEY records. A Key Tag field in the SIG and DS records can be used to help efficiently select the corresponding KEY RR when there is more than one candidate KEY RR 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, same algorithm, and same key tag. The key tag is used to efficiently limit the possible candidate keys but it does not uniquely identify a KEY record. Implementations MUST NOT assume the key tag uniquely idenifies a KEY RR.

The following ANSI C reference implementation is provided for calculating a Key Tag. This reference implementation applies to all algorithm types except algorithm 1 (see Appendix B.1). The input is the public key material in base 64, not the entire RDATA of the KEY record that contains the public key. The code is written 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;
}
```


B.1 Key Tag for Algorithm 1 - RSA/MD5

Algorithm 1 - RSA/MD5 key tag is the only algoritm that does not use the key tag defined above. For a KEY RR with algorithm 1, the key tag is the most signifigant 16 bits of the least signifigant 24 bits

in the public key modulus. In others, the 4th to last and 3rd to last octets in the key modulus. Note that Algorithm 1 is NOT RECOMMENDED.

Appendix C. Canonical Form and Order of Resource Records

This section defines a canonical form for resource records (RRs) and defines a name order and overall order. A canonical name order is required to construct the NXT name chain. A canonical RR form and ordering within an RRset is required to construct and verify SIG RRs.

C.1 Canonical DNS Name Order

For purposes of DNS security, owner names are sorted by treating individual labels as unsigned left justified octet strings. The absence of a octet sorts before a zero value octet and upper case letters are treated as lower case letters.

To sort names in a zone, first sort all names based on only the highest level label. Next if multiple names appear within a level, sort based on the next highest level label. Repeat until all names have been sorted down to leaf node labels.

For example, the following names are sorted in canonical DNS name order. The highest label is label level is foo.example. At this level, foo.example sorts first, followed by all names ending in a.foo.example and then all names ending z.foo.example. The names withing the a.foo.example level and z.foo.example level are sorted.

```
foo.example
a.foo.example
yljkjljk.a.foo.example
Z.a.foo.example
zABC.a.FOO.EXAMPLE
z.foo.example
*.z.foo.example
\200.z.foo.example
```

C.2 Canonical RR Form

For purposes of DNS security, the canonical form for an RR is the wire format of the RR with

(1) all domain names fully expanded
    (no name compression via pointers)
(2) all domain name letters set to lower case
(3) any owner name wild cards in master file form
    (no substitution made for *)
(4) the original TTL substituted for the current TTL.

C.3 Canonical RR Ordering Within An RRset

For purposes of DNS security, RRs with same owner name and same type are sorted by treating the RDATA as a left justified unsigned octet sequence. The absence of an octet sorts before the zero octet.

C.4 Canonical Ordering of RR Types

RRs with the same owner name but different types are sorted based on the RR type number. The exception to this rule are SIG RRs, which are placed immediately after the type they cover.

For example, an A record would be put before an MX record because type 1 (A) and is lower than type 15 (MX). If the A and MX records were both signed, the order would be A < SIG(A) < MX < SIG(MX).

Full Copyright Statement

Acknowledgement

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