DNS Security Working Group Donald E. Eastlake 3rd
INTERNET-DRAFT IBM
OBSOLETES RFC 2065
UPDATES RFCs 1034, 1035, and 2181
Expires: June 1999 December 1998

Domain Name System Security Extensions

Status of This Document

This draft, file name draft-ietf-dnssec-secext2-07.txt, is intended to become a Proposed Standard RFC obsoleting Proposed Standard RFC 2065. Distribution of this document is unlimited. Comments should be sent to the DNS Security Working Group mailing list <dns-security@tis.com> or to the author.

This document is an Internet-Draft. 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. Internet-Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a ``working draft'' or ``work in progress.''

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

[[changed from draft 5 to draft 6: add IANA Considerations section, update dates and file name, update author info, update references where new documents have superceded those previously referenced, add reference to RFC 2119, clarify wording, minor change at 4.1.5 to explain why modular time does not induce a security flaw, clarify that the zone key for a secure subzone need not be included at the leaf node in the superzone, specify that algorithm 254 OIDs are BER encoded, add algorithm 253 for domain name encoded private algorithm]] [[changed from draft 6 to draft 7: tweak IANA Considerations section and add reference to RFC 2434, update author info]]

Abstract

Extensions to the Domain Name System (DNS) are described that provide data integrity and authentication to security aware resolvers and applications through the use of cryptographic digital signatures. These digital signatures are included in secured zones as resource records. Security can also be provided through non-security aware DNS servers in some cases.

The extensions provide for the storage of authenticated public keys in the DNS. This storage of keys can support general public key distribution services as well as DNS security. The stored keys enable security aware resolvers to learn the authenticating key of zones in addition to those for which they are initially configured. Keys associated with DNS names can be retrieved to support other protocols. Provision is made for a variety of key types and algorithms.

In addition, the security extensions provide for the optional authentication of DNS protocol transactions and requests.

This document incorporates feedback on RFC 2065 from early implementers and potential users.

Acknowledgments

The significant contributions and suggestions of the following persons (in alphabetic order) to DNS security are gratefully acknowledged:

James M. Galvin
John Gilmore
Olafur Gudmundsson
Charlie Kaufman
Edward Lewis
Thomas Narten
Radia J. Perlman
Jeffrey I. Schiller
Steven (Xunhua) Wang
Brian Wellington

Table of Contents

Status of This Document.....1

Abstract.....2
Acknowledgments.....2

Table of Contents.....3

1. Overview of Contents.....5
2. Overview of the DNS Extensions.....6
2.1 Services Not Provided.....6
2.2 Key Distribution.....6
2.3 Data Origin Authentication and Integrity.....7
2.3.1 The SIG Resource Record.....8
2.3.2 Authenticating Name and Type Non-existence.....8
2.3.3 Special Considerations With Time-to-Live.....8
2.3.4 Special Considerations at Delegation Points.....9
2.3.5 Special Considerations with CNAME.....9
2.3.6 Signers Other Than The Zone.....10
2.4 DNS Transaction and Request Authentication.....10
3. The KEY Resource Record.....11
3.1 KEY RDATA format.....12
3.1.1 Object Types, DNS Names, and Keys.....12
3.1.2 The KEY RR Flag Field.....13
3.1.3 The Protocol Octet.....14
3.2 The KEY Algorithm Number Specification.....15
3.3 Interaction of Flags, Algorithm, and Protocol Bytes...16
3.4 Determination of Zone Secure/Unsecured Status.....16
3.5 KEY RRs in the Construction of Responses.....18
4. The SIG Resource Record.....18
4.1 SIG RDATA Format.....19
4.1.1 Type Covered Field.....19
4.1.2 Algorithm Number Field.....19
4.1.3 Labels Field.....19
4.1.4 Original TTL Field.....20
4.1.5 Signature Expiration and Inception Fields.....20
4.1.6 Key Tag Field.....21
4.1.7 Signer's Name Field.....21
4.1.8 Signature Field.....22
4.1.8.1 Calculating Transaction and Request SIGs.....22
4.2 SIG RRs in the Construction of Responses.....23
4.3 Processing Responses and SIG RRs.....24
4.4 Signature Lifetime, Expiration, TTLs, and Validity....25
5. Non-existent Names and Types.....25
5.1 The NXT Resource Record.....26
5.2 NXT RDATA Format.....27
5.3 Additional Complexity Due to Wildcards.....27

5.4 Example.....28
5.5 Special Considerations at Delegation Points.....28

5.6 Zone Transfers.....29
5.6.1 Full Zone Transfers.....29
5.6.2 Incremental Zone Transfers.....30
6. How to Resolve Securely and the AD and CD Bits.....30
6.1 The AD and CD Header Bits.....31
6.2 Staticly Configured Keys.....32
6.3 Chaining Through The DNS.....33
6.3.1 Chaining Through KEYs.....33
6.3.2 Conflicting Data.....34
6.4 Secure Time.....35
7. ASCII Representation of Security RRs.....35
7.1 Presentation of KEY RRs.....36
7.2 Presentation of SIG RRs.....37
7.3 Presentation of NXT RRs.....38
8. Canonical Form and Order of Resource Records.....38
8.1 Canonical RR Form.....38
8.2 Canonical DNS Name Order.....38
8.3 Canonical RR Ordering Within An RRset.....39
8.4 Canonical Ordering of RR Types.....39
9. Conformance.....39
9.1 Server Conformance.....39
9.2 Resolver Conformance.....40
10. Security Considerations.....40
11. IANA Considerations.....41

References.....42

Author's Address.....44
Expiration and File Name.....44

Appendix A: Base 64 Encoding.....45

Appendix B: Changes from RFC 2065.....47

Appendix C: Key Tag Calculation.....49

1. Overview of Contents

This document standardizes extensions of the Domain Name System (DNS) protocol to support DNS security and public key distribution. It assumes that the reader is familiar with the Domain Name System, particularly as described in RFCs 1033, 1034, 1035 and later RFCs. An earlier version of these extensions appears in RFC 2065. This replacement for that RFC incorporates early implementation experience and requests from potential users.

Section 2 provides an overview of the extensions and the key distribution, data origin authentication, and transaction and request security they provide.

Section 3 discusses the KEY resource record, its structure, and use in DNS responses. These resource records represent the public keys of entities named in the DNS and are used for key distribution.

Section 4 discusses the SIG digital signature resource record, its structure, and use in DNS responses. These resource records are used to authenticate other resource records in the DNS and optionally to authenticate DNS transactions and requests.

Section 5 discusses the NXT resource record (RR) and its use in DNS responses including full and incremental zone transfers. The NXT RR permits authenticated denial of the existence of a name or of an RR type for an existing name.

Section 6 discusses how a resolver can be configured with a starting key or keys and proceed to securely resolve DNS requests. Interactions between resolvers and servers are discussed for various combinations of security aware and security non-aware. Two additional DNS header bits are defined for signaling between resolvers and servers.

Section 7 describes the ASCII representation of the security resource records for use in master files and elsewhere.

Section 8 defines the canonical form and order of RRs for DNS security purposes.

Section 9 defines levels of conformance for resolvers and servers.

Section 10 provides a few paragraphs on overall security considerations.

Sectuion 11 specified IANA considerations for allocation of additional values of paramters defined in this document.

Appendix A gives details of base 64 encoding which is used in the

file representation of some RRs defined in this document.

Appendix B summarizes changes between this draft and RFC 2065.

Appendix C specified how to calculate the simple checksum used as a key tag in most SIG RRs.

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 [RFC2119].

2. Overview of the DNS Extensions

The Domain Name System (DNS) protocol security extensions provide three distinct services: key distribution as described in Section 2.2 below, data origin authentication as described in Section 2.3 below, and transaction and request authentication, described in Section 2.4 below.

Special considerations related to "time to live", CNAMEs, and delegation points are also discussed in Section 2.3.

2.1 Services Not Provided

It is part of the design philosophy of the DNS that the data in it is public and that the DNS gives the same answers to all inquirers. Following this philosophy, no attempt has been made to include any sort of access control lists or other means to differentiate inquirers.

No effort has been made to provide for any confidentiality for queries or responses. (This service may be available via IPSEC [RFC 1825], TLS, or other security protocols.)

Protection is not provided against denial of service.

2.2 Key Distribution

A resource record format is defined to associate keys with DNS names. This permits the DNS to be used as a public key distribution mechanism in support of DNS security itself and other protocols.

The syntax of a KEY resource record (RR) is described in Section 3. It includes an algorithm identifier, the actual public key

parameter(s), and a variety of flags including those indicating the type of entity the key is associated with and/or asserting that there is no key associated with that entity.

Under conditions described in Section 3.5, security aware DNS servers will automatically attempt to return KEY resources as additional information, along with those resource records actually requested, to minimize the number of queries needed.

2.3 Data Origin Authentication and Integrity

Authentication is provided by associating with resource record sets (RRsets [RFC 2181]) in the DNS cryptographically generated digital signatures. Commonly, there will be a single private key that authenticates an entire zone but there might be multiple keys for different algorithms, signers, etc. If a security aware resolver reliably learns a public key of the zone, it can authenticate, for signed data read from that zone, that it is properly authorized. The most secure implementation is for the zone private key(s) to be kept off-line and used to re-sign all of the records in the zone periodically. However, there are cases, for example dynamic update [RFCs 2136, 2137], where DNS private keys need to be on-line. [draft-ietf-dnssec-secops-*.txt]

The data origin authentication key(s) are associated with the zone and not with the servers that store copies of the data. That means compromise of a secondary server or, if the key(s) are kept off line, even the primary server for a zone, will not necessarily affect the degree of assurance that a resolver has that it can determine whether data is genuine.

A resolver could learn a public key of a zone either by reading it from the DNS or by having it staticly configured. To reliably learn a public key by reading it from the DNS, the key itself must be signed with a key the resolver trusts. The resolver must be configured with at least a public key which authenticates one zone as a starting point. From there, it can securely read public keys of other zones, if the intervening zones in the DNS tree are secure and their signed keys accessible.

Adding data origin authentication and integrity requires no change to the "on-the-wire" DNS protocol beyond the addition of the signature resource type and the key resource type needed for key distribution. (Data non-existence authentication also requires the NXT RR as described in 2.3.2.) This service can be supported by existing resolver and caching server implementations so long as they can support the additional resource types (see Section 9). The one

exception is that CNAME referrals in a secure zone can not be

authenticated if they are from non-security aware servers (see Section 2.3.5).

If signatures are separately retrieved and verified when retrieving the information they authenticate, there will be more trips to the server and performance will suffer. Security aware servers mitigate that degradation by attempting to send the signature(s) needed (see Section 4.2).

2.3.1 The SIG Resource Record

The syntax of a SIG resource record (signature) is described in Section 4. It cryptographicly binds the RRset being signed to the signer and a validity interval.

Every name in a secured zone will have associated with it at least one SIG resource record for each resource type under that name except for glue address RRs and delegation point NS RRs. A security aware server will attempt to return, with RRs retrieved, the corresponding SIGs. If a server is not security aware, the resolver must retrieve all the SIG records for a name and select the one or ones that sign the resource record set(s) that resolver is interested in.

2.3.2 Authenticating Name and Type Non-existence

The above security mechanism only provides a way to sign existing RRsets in a zone. "Data origin" authentication is not obviously provided for the non-existence of a domain name in a zone or the non-existence of a type for an existing name. This gap is filled by the NXT RR which authenticatably asserts a range of non-existent names in a zone and the non-existence of types for the existing name just before that range.

Section 5 below covers the NXT RR.

2.3.3 Special Considerations With Time-to-Live

A digital signature will fail to verify if any change has occurred to the data between the time it was originally signed and the time the signature is verified. This conflicts with our desire to have the time-to-live (TTL) field of resource records tick down while they are cached.

This could be avoided by leaving the time-to-live out of the digital

signature, but that would allow unscrupulous servers to set arbitrarily long TTL values undetected. Instead, we include the "original" TTL in the signature and communicate that data along with the current TTL. Unscrupulous servers under this scheme can manipulate the TTL but a security aware resolver will bound the TTL value it uses at the original signed value. Separately, signatures include a signature inception time and a signature expiration time. A resolver that knows the absolute time can determine securely whether a signature is in effect. It is not possible to rely solely on the signature expiration as a substitute for the TTL, however, since the TTL is primarily a database consistency mechanism and non-security aware servers that depend on TTL must still be supported.

2.3.4 Special Considerations at Delegation Points

DNS security would like to view each zone as a unit of data completely under the control of the zone owner with each entry (RRset) signed by a special private key held by the zone manager. But the DNS protocol views the leaf nodes in a zone, which are also the apex nodes of a subzone (i.e., delegation points), as "really" belonging to the subzone. These nodes occur in two master files and might have RRs signed by both the upper and lower zone's keys. A retrieval could get a mixture of these RRs and SIGs, especially since one server could be serving both the zone above and below a delegation point. [RFC 2181]

There MUST be a zone KEY RR, signed by its superzone, for every subzone if the superzone is secure. This will normally appear in the subzone and may also be included in the superzone. But, in the case of an unsecured subzone which can not or will not be modified to add any security RRs, a KEY declaring the subzone to be unsecured MUST appear with the superzone signature in the superzone, if the superzone is secure. For all but one other RR type the data from the subzone is more authoritative so only the subzone KEY RR should be signed in the superzone if it appears there. The NS and any glue address RRs SHOULD only be signed in the subzone. The SOA and any other RRs that have the zone name as owner should appear only in the subzone and thus are signed only there. The NXT RR type is the exceptional case that will always appear differently and authoritatively in both the superzone and subzone, if both are secure, as described in Section 5.

2.3.5 Special Considerations with CNAME

There is a problem when security related RRs with the same owner name

as a CNAME RR are retrieved from a non-security-aware server. In

particular, an initial retrieval for the CNAME or any other type may not retrieve any associated SIG, KEY, or NXT RR. For retrieved types other than CNAME, it will retrieve that type at the target name of the CNAME (or chain of CNAMEs) and will also return the CNAME. In particular, a specific retrieval for type SIG will not get the SIG, if any, at the original CNAME domain name but rather a SIG at the target name.

Security aware servers must be used to securely CNAME in DNS. Security aware servers MUST (1) allow KEY, SIG, and NXT RRs along with CNAME RRs, (2) suppress CNAME processing on retrieval of these types as well as on retrieval of the type CNAME, and (3) automatically return SIG RRs authenticating the CNAME or CNAMEs encountered in resolving a query. This is a change from the previous DNS standard [RFCs 1034/1035] which prohibited any other RR type at a node where a CNAME RR was present.

2.3.6 Signers Other Than The Zone

There are cases where the signer in a SIG resource record is other than one of the private key(s) used to authenticate a zone.

One is for support of dynamic update [RFC 2136] (or future requests which require secure authentication) where an entity is permitted to authenticate/update its records [RFC 2137] and the zone is operating in a mode where the zone key is not on line. The public key of the entity must be present in the DNS and be signed by a zone level key but the other RR(s) may be signed with the entity's key.

A second case is support of transaction and request authentication as described in Section 2.4.

In additions, signatures can be included on resource records within the DNS for use by applications other than DNS. DNS related signatures authenticate that data originated with the authority of a zone owner or that a request or transaction originated with the relevant entity. Other signatures can provide other types of assurances.

2.4 DNS Transaction and Request Authentication

The data origin authentication service described above protects retrieved resource records and the non-existence of resource records but provides no protection for DNS requests or for message headers.

If header bits are falsely set by a bad server, there is little that

can be done. However, it is possible to add transaction authentication. Such authentication means that a resolver can be sure it is at least getting messages from the server it thinks it queried and that the response is from the query it sent (i.e., that these messages have not been diddled in transit). This is accomplished by optionally adding a special SIG resource record at the end of the reply which digitally signs the concatenation of the server's response and the resolver's query.

Requests can also be authenticated by including a special SIG RR at the end of the request. Authenticating requests serves no function in older DNS servers and requests with a non-empty additional information section produce error returns or may even be ignored by many of them. However, this syntax for signing requests is defined as a way of authenticating secure dynamic update requests [RFC 2137] or future requests requiring authentication.

The private keys used in transaction security belong to the entity composing the reply, not to the zone involved. Request authentication may also involve the private key of the host or other entity composing the request or other private keys depending on the request authority it is sought to establish. The corresponding public key(s) are normally stored in and retrieved from the DNS for verification.

Because requests and replies are highly variable, message authentication SIGs can not be pre-calculated. Thus it will be necessary to keep the private key on-line, for example in software or in a directly connected piece of hardware.

3. The KEY Resource Record

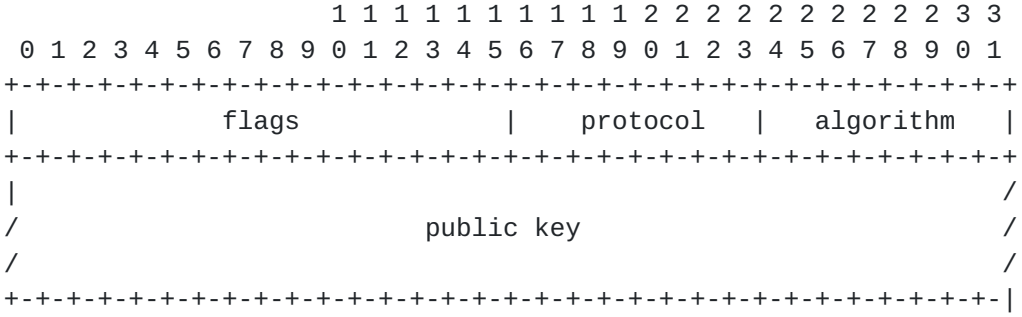
The KEY resource record (RR) is used to store a public key that is associated with a Domain Name System (DNS) name. This can be the public key of a zone, a user, or a host or other end entity. Security aware DNS implementations MUST be designed to handle at least two simultaneously valid keys of the same type associated with the same name.

The type number for the KEY RR is 25.

A KEY RR is, like any other RR, authenticated by a SIG RR. KEY RRs must be signed by a zone level key.

3.1 KEY RDATA format

The RDATA for a KEY RR consists of flags, a protocol octet, the algorithm number octet, and the public key itself. The format is as follows:



The KEY RR is not intended for storage of certificates and a separate certificate RR has been developed for that purpose, defined in [draft-ietf-dnssec-certs-*.txt].

The meaning of the KEY RR owner name, flags, and protocol octet are described in Sections 3.1.1 through 3.1.5 below. The flags and algorithm must be examined before any data following the algorithm octet as they control the existence and format of any following data. The algorithm and public key fields are described in Section 3.2. The format of the public key is algorithm dependent.

KEY RRs do not specify their validity period but their authenticating SIG RR(s) do as described in Section 4 below.

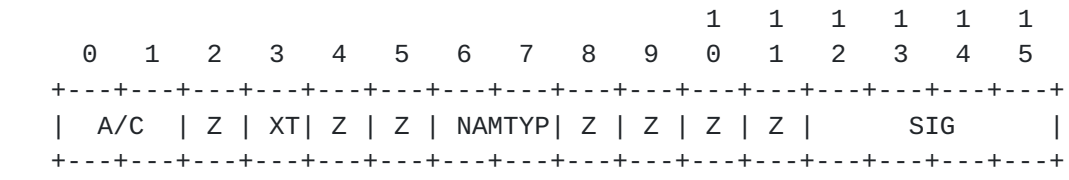
3.1.1 Object Types, DNS Names, and Keys

The public key in a KEY RR is for the object named in the owner name.

A DNS name may refer to three different categories of things. For example, foo.host.example could be (1) a zone, (2) a host or other end entity , or (3) the mapping into a DNS name of the user or account foo@host.example. Thus, there are flag bits, as described below, in the KEY RR to indicate with which of these roles the owner name and public key are associated. Note that an appropriate zone KEY RR MUST occur at the apex node of a secure zone and zone KEY RRs occur only at delegation points.

3.1.2 The KEY RR Flag Field

In the "flags" field:



Bit 0 and 1 are the key "type" bits whose values have the following meanings:

10: Use of the key is prohibited for authentication.

01: Use of the key is prohibited for confidentiality.

00: Use of the key for authentication and/or confidentiality is permitted. Note that DNS security makes use of keys for authentication only. Confidentiality use flagging is provided for use of keys in other protocols. Implementations not intended to support key distribution for confidentiality MAY require that the confidentiality use prohibited bit be on for keys they serve.

11: If both bits are one, the "no key" value, there is no key information and the RR stops after the algorithm octet. By the use of this "no key" value, a signed KEY RR can authenticatably assert that, for example, a zone is not secured. See section 3.4 below.

Bits 2 is reserved and must be zero.

Bits 3 is reserved as a flag extension bit. If it is a one, a second 16 bit flag field is added after the algorithm octet and before the key data. This bit MUST NOT be set unless one or more such additional bits have been defined and are non-zero.

Bits 4-5 are reserved and must be zero.

Bits 6 and 7 form a field that encodes the name type. Field values have the following meanings:

00: indicates that this is a key associated with a "user" or "account" at an end entity, usually a host. The coding of the owner name is that used for the responsible individual mailbox in the SOA and RP RRs: The owner name is the user name as the name of a node under the entity name. For example, "j_random_user" on host.subdomain.example could have a public key associated through a KEY RR with name j_random_user.host.subdomain.example. It could be used in a security protocol where authentication of a user was desired. This key might be useful in IP or other security for a user level service such a telnet, ftp, rlogin, etc.

01: indicates that this is a zone key for the zone whose name

is the KEY RR owner name. This is the public key used for the primary DNS security feature of data origin authentication. Zone

KEY RRs occur only at delegation points.

10: indicates that this is a key associated with the non-zone "entity" whose name is the RR owner name. This will commonly be a host but could, in some parts of the DNS tree, be some other type of entity such as a telephone number [RFC 1530] or numeric IP address. This is the public key used in connection with DNS request and transaction authentication services. It could also be used in an IP-security protocol where authentication at the host, rather than user, level was desired, such as routing, NTP, etc.

11: reserved.

Bits 8-11 are reserved and must be zero.

Bits 12-15 are the "signatory" field. If non-zero, they indicate that the key can validly sign things as specified in DNS dynamic update [RFC 2137]. Note that zone keys (see bits 6 and 7 above) always have authority to sign any RRs in the zone regardless of the value of the signatory field.

3.1.3 The Protocol Octet

It is anticipated that keys stored in DNS will be used in conjunction with a variety of Internet protocols. It is intended that the protocol octet and possibly some of the currently unused (must be zero) bits in the KEY RR flags as specified in the future will be used to indicate a key's validity for different protocols.

The following values of the Protocol Octet are reserved as indicated:

VALUE	Protocol
0	-reserved
1	TLS
2	email
3	dnssec
4	IPSEC
5-254	- available for assignment by IANA
255	All

In more detail:

1 is reserved for use in connection with TLS.

2 is reserved for use in connection with email.

3 is used for DNS security. The protocol field SHOULD be set to this value for zone keys and other keys used in DNS security. Implementations that can determine that a key is a DNS security key by the fact that flags label it a zone key or the signatory flag field is non-zero are NOT REQUIRED to check the protocol field.

4 is reserved to refer to the Oakley/IPSEC [RFC 1825] protocol

and indicates that this key is valid for use in conjunction with that security standard. This key could be used in connection with secured communication on behalf of an end entity or user whose name is the owner name of the KEY RR if the entity or user flag bits are set. The presence of a KEY resource with this protocol value is an assertion that the host speaks Oakley/IPSEC.

255 indicates that the key can be used in connection with any protocol for which KEY RR protocol octet values have been defined. The use of this value is discouraged and the use of different keys for different protocols is encouraged.

3.2 The KEY Algorithm Number Specification

This octet is the key algorithm parallel to the same field for the SIG resource as described in Section 4.1. The following values are assigned:

VALUE	Algorithm
0	- reserved, see Section 11
1	RSA/MD5 [draft-ietf-dnssec-rsa-*.txt] - recommended
2	Diffie-Hellman [draft-ietf-dnssec-dhk-*.txt] - optional, key only
3	DSA [draft-ietf-dnssec-dss-*.txt] - MANDATORY
4	reserved for elliptic curve crypto
5-251	- available, see Section 11
252	reserved for indirect keys
253	private - domain name (see below)
254	private - OID (see below)
255	- reserved, see Section 11

Algorithm specific formats and procedures are given in separate documents. The mandatory to implement for interoperability algorithm is number 3, DSA. It is recommended that the RSA/MD5 algorithm, number 1, also be implemented. Algorithm 2 is used to indicate Diffie-Hellman keys and algorithm 4 is reserved for elliptic curve.

Algorithm number 252 indicates an indirect key format where the actual key material is elsewhere. This format is to 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. 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 whatever is required by that algorithm. 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.

Values 0 and 255 are reserved but the value 0 is used in the algorithm field when that field is not used. An example is in a KEY RR with the top two flag bits on, the "no-key" value, where no key is present.

3.3 Interaction of Flags, Algorithm, and Protocol Bytes

Various combinations of the no-key type flags, algorithm byte, protocol byte, and any future assigned protocol indicating flags are possible. The meaning of these combinations is indicated below:

NK = no key type (flags bits 0 and 1 on)
AL = algorithm byte
PR = protocols indicated by protocol byte or future assigned flags

x represents any valid non-zero value(s).

AL	PR	NK	Meaning
0	0	0	Illegal, claims key but has bad algorithm field.
0	0	1	Specifies total lack of security for owner zone.
0	x	0	Illegal, claims key but has bad algorithm field.
0	x	1	Specified protocols unsecured, others may be secure.
x	0	0	Gives key but no protocols to use it.
x	0	1	Denies key for specific algorithm.
x	x	0	Specifies key for protocols.
x	x	1	Algorithm not understood for protocol.

3.4 Determination of Zone Secure/Unsecured Status

A zone KEY RR with the "no-key" type field value (both key type flag bits 0 and 1 on) indicates that the zone named is unsecured while a zone KEY RR with a key present indicates that the zone named is secure. The secured versus unsecured status of a zone may vary with different cryptographic algorithms. Even for the same algorithm, conflicting zone KEY RRs may be present.

Zone KEY RRs, like all RRs, are only trusted if they are authenticated by a SIG RR whose signer field is a signer for which the resolver has a public key they trust and where resolver policy permits that signer to sign for the KEY owner name. Untrusted zone

KEY RRs MUST be ignored in determining the security status of the

zone. However, there can be multiple sets of trusted zone KEY RRs for a zone with different algorithms, signers, etc.

For any particular algorithm, zones can be (1) secure, indicating that any retrieved RR must be authenticated by a SIG RR or it will be discarded as bogus, (2) unsecured, indicating that SIG RRs are not expected or required for RRs retrieved from the zone, or (3) experimentally secure, which indicates that SIG RRs might or might not be present but must be checked if found. The status of a zone is determined as follows:

1. If, for a zone and algorithm, every trusted zone KEY RR for the zone says there is no key for that zone, it is unsecured for that algorithm.

2. If, there is at least one trusted no-key zone KEY RR and one trusted key specifying zone KEY RR, then that zone is only experimentally secure for the algorithm. Both authenticated and non-authenticated RRs for it should be accepted by the resolver.

3. If every trusted zone KEY RR that the zone and algorithm has is key specifying, then it is secure for that algorithm and only authenticated RRs from it will be accepted.

Examples:

(1) A resolver initially trusts only signatures by the superzone of zone Z within the DNS hierarchy. Thus it will look only at the KEY RRs that are signed by the superzone. If it finds only no-key KEY RRs, it will assume the zone is not secure. If it finds only key specifying KEY RRs, it will assume the zone is secure and reject any unsigned responses. If it finds both, it will assume the zone is experimentally secure

(2) A resolver trusts the superzone of zone Z (to which it got securely from its local zone) and a third party, cert-auth.example. When considering data from zone Z, it may be signed by the superzone of Z, by cert-auth.example, by both, or by neither. The following table indicates whether zone Z will be considered secure, experimentally secure, or unsecured, depending on the signed zone KEY RRs for Z;

c e r t - a u t h . e x a m p l e

Superzone	KEY RRs: None	NoKeys	Mixed	Keys
None	illegal	unsecured	experim.	secure
NoKeys	unsecured	unsecured	experim.	secure
Mixed	experim.	experim.	experim.	secure
Keys	secure	secure	secure	secure

3.5 KEY RRs in the Construction of Responses

An explicit request for KEY RRs does not cause any special additional information processing except, of course, for the corresponding SIG RR from a security aware server (see Section 4.2).

Security aware DNS servers include KEY RRs as additional information in responses, where a KEY is available, in the following cases:

(1) On the retrieval of SOA or NS RRs, the KEY RRset with the same name (perhaps just a zone key) SHOULD be included as additional information if space is available. If not all additional information will fit, type A and AAAA glue RRs have higher priority than KEY RR(s).

(2) On retrieval of type A or AAAA RRs, the KEY RRset with the same name (usually just a host RR and NOT the zone key (which usually would have a different name)) SHOULD be included if space is available. On inclusion of A or AAAA RRs as additional information, the KEY RRset with the same name should also be included but with lower priority than the A or AAAA RRs.

4. 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.

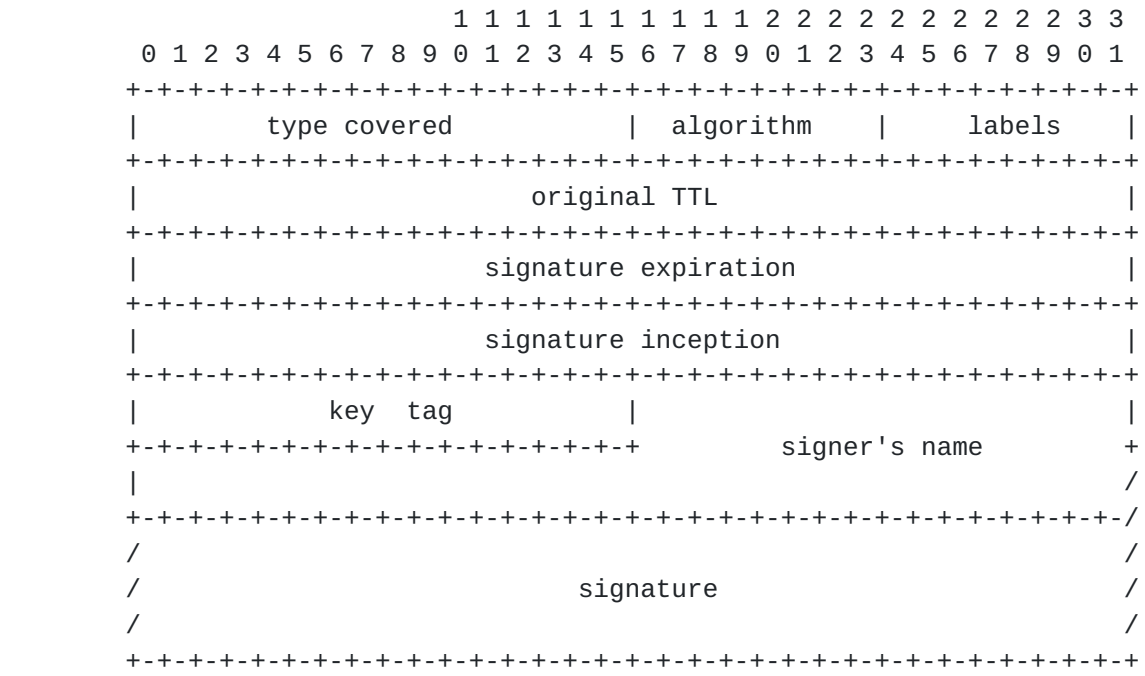
The SIG RR unforgably authenticates an RRset [RFC 2181] of a particular type, class, and name and binds it to a time interval and the signer's domain name. This is done using cryptographic techniques and the signer's private key. The signer is frequently

the owner of the zone from which the RR originated.

The type number for the SIG RR type is 24.

4.1 SIG RDATA Format

The RDATA portion of a SIG RR is as shown below. The integrity of the RDATA information is protected by the signature field.



4.1.1 Type Covered Field

The "type covered" is the type of the other RRs covered by this SIG.

4.1.2 Algorithm Number Field

This octet is as described in section 3.2.

4.1.3 Labels Field

The "labels" octet is an unsigned count of how many labels there are in the original SIG RR owner name not counting the null label for root and not counting any initial "*" for a wildcard. If a secured retrieval is the result of wild card substitution, it is necessary

for the resolver to use the original form of the name in verifying

the digital signature. This field makes it easy to determine the original form.

If, on retrieval, the RR appears to have a longer name than indicated by "labels", the resolver can tell it is the result of wildcard substitution. If the RR owner name appears to be shorter than the labels count, the SIG RR must be considered corrupt and ignored. The maximum number of labels allowed in the current DNS is 127 but the entire octet is reserved and would be required should DNS names ever be expanded to 255 labels. The following table gives some examples. The value of "labels" is at the top, the retrieved owner name on the left, and the table entry is the name to use in signature verification except that "bad" means the RR is corrupt.

labels=	0	1	2	3	4
.	.	bad	bad	bad	bad
d.	*.	d.	bad	bad	bad
c.d.	*.	*.d.	c.d.	bad	bad
b.c.d.	*.	*.d.	*.c.d.	b.c.d.	bad
a.b.c.d.	*.	*.d.	*.c.d.	*.b.c.d.	a.b.c.d.

4.1.4 Original TTL Field

The "original TTL" field is included in the RDATA portion to avoid (1) authentication problems that caching servers would otherwise cause by decrementing the real TTL field and (2) security problems that unscrupulous servers could otherwise cause by manipulating the real TTL field. This original TTL is protected by the signature while the current TTL field is not.

NOTE: The "original TTL" must be restored into the covered RRs when the signature is verified (see Section 8). This generaly implies that all RRs for a particular type, name, and class, that is, all the RRs in any particular RRset, must have the same TTL to start with.

4.1.5 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. (See also Section 4.4.) Ring arithmetic is used as for DNS SOA serial numbers [RFC 1982] 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. However there is no

security flaw because keys are required to be changed to new random

keys by [draft-ietf-dnssec-secops-*.txt] at least every five years. This means that the probability that the same key is in use N*136.09 years later should be the same as the probability that a random guess will work.

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.

(To prevent misordering of network requests to update a zone dynamically, monotonically increasing "signature inception" times may be necessary.)

A secure zone must be considered changed for SOA serial number purposes not only when its data is updated but also when new SIG RRs are inserted (ie, the zone or any part of it is re-signed).

4.1.6 Key Tag Field

The "key Tag" is a two octet quantity that is used to efficiently select between multiple keys which may be applicable and thus check that a public key about to be used for the computationally expensive effort to check the signature is possibly valid. For algorithm 1 (MD5/RSA) as defined in [draft-ietf-dnssec-rsa-*.txt], it is the next to the bottom two octets of the public key modulus needed to decode the signature field. That is to say, the most significant 16 of the least significant 24 bits of the modulus in network (big endian) order. For all other algorithms, including private algorithms, it is calculated as a simple checksum of the KEY RR as described in Appendix C.

4.1.7 Signer's Name Field

The "signer's name" field is the domain name of the signer generating the SIG RR. This is the owner name of the public KEY RR that can be used to verify the signature. It is frequently the zone which contained the RRset being authenticated. Which signers should be authorized to sign what is a significant resolver policy question as discussed in Section 6. The signer's name may be compressed with standard DNS name compression when being transmitted over the network.

4.1.8 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. This covered RRset is thereby authenticated. To accomplish this, a data sequence is constructed as follows:

```
data = RDATA | RR(s)...
```

where "|" is concatenation,

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(s) is the RRset of the RR(s) of the type covered with the same owner name and class as the SIG RR in canonical form and order as defined in Section 8.

How this data sequence is processed into the signature is algorithm dependent. These algorithm dependent formats and procedures are described in separate documents (Section 3.2).

SIGs SHOULD NOT be included in a zone for any "meta-type" such as ANY, AXFR, etc. (but see section 5.6.2 with regard to IXFR).

4.1.8.1 Calculating Transaction and Request SIGs

A response message from a security aware server may optionally contain a special SIG at the end of the additional information section to authenticate the transaction.

This SIG has a "type covered" field of zero, which is not a valid RR type. It is calculated by using a "data" (see Section 4.1.8) of the entire preceding DNS reply message, including DNS header but not the IP header and before the reply RR counts have been adjusted for the inclusion of any transaction SIG, concatenated with the entire DNS query message that produced this response, including the query's DNS header and any request SIGs but not its IP header. That is

```
data = full response (less transaction SIG) | full query
```

Verification of the transaction SIG (which is signed by the server host key, not the zone key) by the requesting resolver shows that the query and response were not tampered with in transit, that the response corresponds to the intended query, and that the response comes from the queried server.

A DNS request may be optionally signed by including one or more SIGs at the end of the query. Such SIGs are identified by having a "type covered" field of zero. They sign the preceding DNS request message including DNS header but not including the IP header or any request SIGs at the end and before the request RR counts have been adjusted for the inclusions of any request SIG(s).

WARNING: Request SIGs are unnecessary for any currently defined request other than update [RFC 2136, 2137] and will cause some old DNS servers to give an error return or ignore a query. However, such SIGs may in the future be needed for other requests.

Except where needed to authenticate an update or similar privileged request, servers are not required to check request SIGs.

4.2 SIG RRs in the Construction of Responses

Security aware DNS servers SHOULD, for every authenticated RRset the query will return, attempt to send the available SIG RRs which authenticate the requested RRset. The following rules apply to the inclusion of SIG RRs in responses:

1. when an RRset is placed in a response, its SIG RR has a higher priority for inclusion than additional RRs that may need to be included. If space does not permit its inclusion, the response MUST be considered truncated except as provided in 2 below.

2. When a SIG RR is present in the zone for an additional information section RR, the response MUST NOT be considered truncated merely because space does not permit the inclusion of the SIG RR with the additional information.

3. SIGs to authenticate glue records and NS RRs for subzones at a delegation point are unnecessary and MUST NOT be sent.

4. If a SIG covers any RR that would be in the answer section of the response, its automatic inclusion MUST be in the answer section. If it covers an RR that would appear in the authority section, its automatic inclusion MUST be in the authority section. If it covers an RR that would appear in the additional information section it MUST appear in the additional information section. This is a change in the existing standard [RFCs 1034, 1035] which contemplates only NS and SOA RRs in the authority section.

5. Optionally, DNS transactions may be authenticated by a SIG RR at the end of the response in the additional information section

(Section 4.1.8.1). Such SIG RRs are signed by the DNS server

originating the response. Although the signer field MUST be a name of the originating server host, the owner name, class, TTL, and original TTL, are meaningless. The class and TTL fields SHOULD be zero. To conserve space, the owner name SHOULD be root (a single zero octet). If transaction authentication is desired, that SIG RR must be considered the highest priority for inclusion.

4.3 Processing Responses and SIG RRs

The following rules apply to the processing of SIG RRs included in a response:

1. A security aware resolver that receives a response from a security aware server via a secure communication with the AD bit (see Section 6.1) set, MAY choose to accept the RRs as received without verifying the zone SIG RRs.

2. In other cases, a security aware resolver SHOULD verify the SIG RRs for the RRs of interest. This may involve initiating additional queries for SIG or KEY RRs, especially in the case of getting a response from a server that does not implement security. (As explained in 2.3.5 above, it will not be possible to secure CNAMEs being served up by non-secure resolvers.)

   NOTE: Implementers might expect the above SHOULD to be a MUST. However, local policy or the calling application may not require the security services.

3. If SIG RRs are received in response to a user query explicitly specifying the SIG type, no special processing is required.

If the message does not pass integrity checks or the SIG does not check against the signed RRs, the SIG RR is invalid and should be ignored. If all of the SIG RR(s) purporting to authenticate an RRset are invalid, then the RRset is not authenticated.

If the SIG RR is the last RR in a response in the additional information section and has a type covered of zero, it is a transaction signature of the response and the query that produced the response. It MAY be optionally checked and the message rejected if the checks fail. But even if the checks succeed, such a transaction authentication SIG does NOT directly authenticate any RRs in the message. Only a proper SIG RR signed by the zone or a key tracing its authority to the zone or to static resolver configuration can directly authenticate RRs, depending on resolver policy (see Section 6). If a resolver does not implement transaction and/or request

SIGs, it MUST ignore them without error.

If all checks indicate that the SIG RR is valid then RRs verified by it should be considered authenticated.

4.4 Signature Lifetime, Expiration, TTLs, and Validity

Security aware servers MUST NOT consider SIG RRs to authenticate anything before their signature inception or after its expiration time (see also Section 6). Security aware servers MUST NOT consider any RR to be authenticated after all its signatures have expired. When a secure server caches authenticated data, if the TTL would expire at a time further in the future than the authentication expiration time, the server SHOULD trim the TTL in the cache entry not to extent beyond the authentication expiration time. Within these constraints, servers should continue to follow DNS TTL aging. Thus authoritative servers should continue to follow the zone refresh and expire parameters and a non-authoritative server should count down the TTL and discard RRs when the TTL is zero (even for a SIG that has not yet reached its authentication expiration time). In addition, when RRs are transmitted in a query response, the TTL should be trimmed so that current time plus the TTL does not extend beyond the authentication expiration time. Thus, in general, the TTL on a transmitted RR would be

```
min(authExpTim,max(zoneMinTTL,min(originalTTL,currentTTL)))
```

When signatures are generated, signature expiration times should be set far enough in the future that it is quite certain that new signatures can be generated before the old ones expire. However, setting expiration too far into the future could mean a long time to flush any bad data or signatures that may have been generated.

It is recommended that signature lifetime be a small multiple of the TTL (ie, 4 to 16 times the TTL) but not less than a reasonable maximum re-signing interval and not less than the zone expiry time.

5. Non-existent Names and Types

The SIG RR mechanism described in Section 4 above provides strong authentication of RRs that exist in a zone. But it is not clear above how to verifiably deny the existence of a name in a zone or a type for an existent name.

The nonexistence of a name in a zone is indicated by the NXT ("next") RR for a name interval containing the nonexistent name. An NXT RR or RRs and its or their SIG(s) are returned in the authority section,

along with the error, if the server is security aware. The same is

true for a non-existent type under an existing name except that there is no error indication other than an empty answer section accompanying the NXT(s). This is a change in the existing standard [RFCs 1034/1035] which contemplates only NS and SOA RRs in the authority section. NXT RRs will also be returned if an explicit query is made for the NXT type.

The existence of a complete set of NXT records in a zone means that any query for any name and any type to a security aware server serving the zone will result in an reply containing at least one signed RR unless it is a query for delegation point NS or glue A or AAAA RRs.

5.1 The NXT Resource Record

The NXT resource record is used to securely indicate that RRs with an owner name in a certain name interval do not exist in a zone and to indicate what RR types are present for an existing name.

The owner name of the NXT RR is an existing name in the zone. It's RDATA is a "next" name and a type bit map. Thus the NXT RRs in a zone create a chain of all of the literal owner names in that zone, including unexpanded wildcards but omitting the owner name of glue address records unless they would otherwise be included. This implies a canonical ordering of all domain names in a zone as described in Section 8. The presence of the NXT RR means that no name between its owner name and the name in its RDATA area exists and that no other types exist under its owner name.

There is a potential problem with the last NXT in a zone as it wants to have an owner name which is the last existing name in canonical order, which is easy, but it is not obvious what name to put in its RDATA to indicate the entire remainder of the name space. This is handled by treating the name space as circular and putting the zone name in the RDATA of the last NXT in a zone.

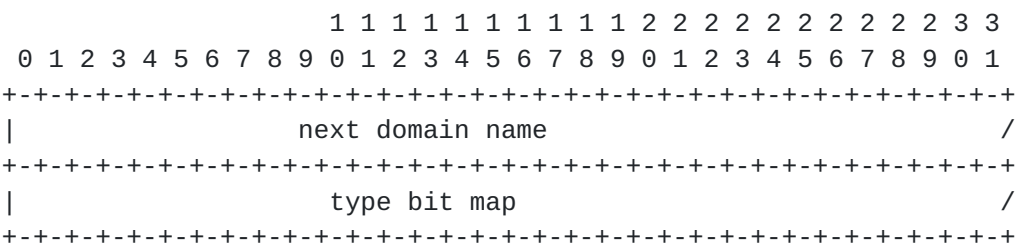
The NXT RRs for a zone SHOULD be automatically calculated and added to the zone when SIGs are added. The NXT RR's TTL SHOULD NOT exceed the zone minimum TTL.

The type number for the NXT RR is 30.

NXT RRs are only signed by zone level keys.

5.2 NXT RDATA Format

The RDATA for an NXT RR consists simply of a domain name followed by a bit map, as shown below.



The NXT RR type bit map format currently defined is one bit per RR type present for the owner name. A one bit indicates that at least one RR of that type is present for the owner name. A zero indicates that no such RR is present. All bits not specified because they are beyond the end of the bit map are assumed to be zero. Note that bit 30, for NXT, will always be on so the minimum bit map length is actually four octets. Trailing zero octets are prohibited in this format. The first bit represents RR type zero (an illegal type which can not be present) and so will be zero in this format. This format is not used if there exists an RR with a type number greater than 127. If the zero bit of the type bit map is a one, it indicates that a different format is being used which will always be the case if a type number greater than 127 is present.

The domain name may be compressed with standard DNS name compression when being transmitted over the network. The size of the bit map can be inferred from the RDLENGTH and the length of the next domain name.

5.3 Additional Complexity Due to Wildcards

Proving that a non-existent name response is correct or that a wildcard expansion response is correct makes things a little more complex.

In particular, when a non-existent name response is returned, an NXT must be returned showing that the exact name queried did not exist and, in general, one or more additional NXT's need to be returned to also prove that there wasn't a wildcard whose expansion should have been returned. (There is no need to return multiple copies of the same NXT.) These NXTs, if any, are returned in the authority section of the response.

Furthermore, if a wildcard expansion is returned in a response, in general one or more NXTs needs to also be returned in the authority

section to prove that no more specific name (including possibly more

specific wildcards in the zone) existed on which the response should have been based.

5.4 Example

Assume zone foo.nil has entries for

big.foo.nil,
medium.foo.nil.
small.foo.nil.
tiny.foo.nil.

Then a query to a security aware server for huge.foo.nil would produce an error reply with an RCODE of NXDOMAIN and the authority section data including something like the following:

```
foo.nil.    NXT big.foo.nil NS KEY SOA NXT ;prove no *.foo.nil
foo.nil.    SIG NXT 1 2 ( ;type-cov=NXT, alg=1, labels=2
                 19970102030405 ;signature expiration
                 19961211100908 ;signature inception
                 2143           ;key identifier
                 foo.nil.       ;signer
AIYADP8d3zYNyQwW2EM4wXVFdslEJcUx/fxkfBeH1El4ixPFhpfHFElxbvKoWmvjDTCm
fiYy2X+8XpFjwICHc398kzWsTMKlxovpz2FnCTM= ;signature (640 bits)
                       )
big.foo.nil. NXT medium.foo.nil. A MX SIG NXT ;prove no huge.foo.nil
big.foo.nil. SIG NXT 1 3 ( ;type-cov=NXT, alg=1, labels=3
                 19970102030405 ;signature expiration
                 19961211100908 ;signature inception
                 2143           ;key identifier
                 foo.nil.       ;signer
 MxFcby9k/yvedMfQgKzhH5er0Mu/vILz45IkskceFGgiWCn/GxHhai6VAuHAoNUz4YoU
 1tVfSCSqQYn6//11U6Nld80jEeC8aTrO+KKmCaY= ;signature (640 bits)
                       )
```

Note that this response implies that big.foo.nil is an existing name in the zone and thus has other RR types associated with it than NXT. However, only the NXT (and its SIG) RR appear in the response to this query for huge.foo.nil, which is a non-existent name.

5.5 Special Considerations at Delegation Points

A name (other than root) which is the head of a zone also appears as the leaf in a superzone. If both are secure, there will always be two different NXT RRs with the same name. They can be easily

distinguished by their signers, the next domain name fields, the

presence of the SOA type bit, etc. Security aware servers should return the correct NXT automatically when required to authenticate the non-existence of a name and both NXTs, if available, on explicit query for type NXT.

Non-security aware servers will never automatically return an NXT and some old implementations may only return the NXT from the subzone on explicit queries.

5.6 Zone Transfers

The subsections below describe how full and incremental zone transfers are secured.

SIG RRs secure all authoritative RRs transferred for both full and incremental [RFC 1995] zone transfers. NXT RRs are an essential element in secure zone transfers and assure that every authoritative name and type will be present; however, if there are multiple SIGs with the same name and type covered, a subset of the SIGs could be sent as long as at least one is present and, in the case of unsigned delegation point NS or glue A or AAAA RRs a subset of these RRs or simply a modified set could be sent as long as at least one of each type is included.

When an incremental or full zone transfer request is received with the same or newer version number than that of the server's copy of the zone, it is replied to with just the SOA RR of the server's current version and the SIG RRset verifying that SOA RR.

The complete NXT chains specified in this document enable a resolver to obtain, by successive queries chaining through NXTs, all of the names in a zone even if zone transfers are prohibited. Different format NXTs may be specified in the future to avoid this.

5.6.1 Full Zone Transfers

To provide server authentication that a complete transfer has occurred, transaction authentication SHOULD be used on full zone transfers. This provides strong server based protection for the entire zone in transit.

5.6.2 Incremental Zone Transfers

Individual RRs in an incremental (IXFR) transfer [RFC 1995] can be verified in the same way as for a full zone transfer and the integrity of the NXT name chain and correctness of the NXT type bits for the zone after the incremental RR deletes and adds can check each disjoint area of the zone updated. But the completeness of an incremental transfer can not be confirmed because usually neither the deleted RR section nor the added RR section has a compete zone NXT chain. As a result, a server which securely supports IXFR must handle IXFR SIG RRs for each incremental transfer set that it maintains.

The IXFR SIG is calculated over the incremental zone update collection of RRs in the order in which it is transmitted: old SOA, then deleted RRs, then new SOA and added RRs. Within each section, RRs must be ordered as specified in Section 8. If condensation of adjacent incremental update sets is done by the zone owner, the original IXFR SIG for each set included in the condensation must be discarded and a new on IXFR SIG calculated to cover the resulting condensed set.

The IXFR SIG really belongs to the zone as a whole, not to the zone name. Although it SHOULD be correct for the zone name, the labels field of an IXFR SIG is otherwise meaningless. The IXFR SIG is only sent as part of an incremental zone transfer. After validation of the IXFR SIG, the transferred RRs MAY be considered valid without verification of the internal SIGs if such trust in the server conforms to local policy.

6. How to Resolve Securely and the AD and CD Bits

Retrieving or resolving secure data from the Domain Name System (DNS) involves starting with one or more trusted public keys that have been staticly configured at the resolver. With starting trusted keys, a resolver willing to perform cryptography can progress securely through the secure DNS structure to the zone of interest as described in Section 6.3. Such trusted public keys would normally be configured in a manner similar to that described in Section 6.2. However, as a practical matter, a security aware resolver would still gain some confidence in the results it returns even if it was not configured with any keys but trusted what it got from a local well known server as if it were staticly configured.

Data stored at a security aware server needs to be internally categorized as Authenticated, Pending, or Insecure. There is also a fourth transient state of Bad which indicates that all SIG checks

have explicitly failed on the data. Such Bad data is not retained at

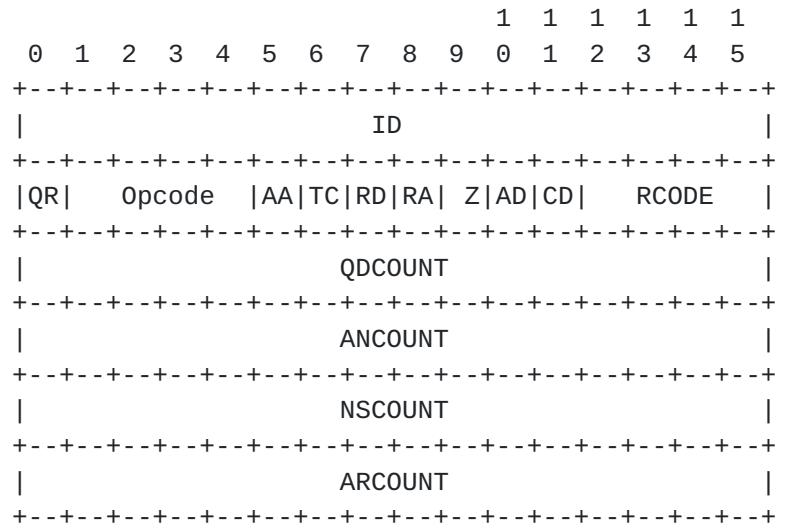
a security aware server. Authenticated means that the data has a valid SIG under a KEY traceable via a chain of zero or more SIG and KEY RRs allowed by the resolvers policies to a KEY staticly configured at the resolver. Pending data has no authenticated SIGs and at least one additional SIG the resolver is still trying to authenticate. Insecure data is data which it is known can never be either Authenticated or found Bad in the zone where it was found because it is in or has been reached via a unsecured zone or because it is unsigned glue address or delegation point NS data. Behavior in terms of control of and flagging based on such data labels is described in Section 6.1.

The proper validation of signatures requires a reasonably secure shared opinion of the absolute time between resolvers and servers as described in Section 6.4.

6.1 The AD and CD Header Bits

Two previously unused bits are allocated out of the DNS query/response format header. The AD (authentic data) bit indicates in a response that all the data included in the answer and authority portion of the response has been authenticated by the server according to the policies of that server. The CD (checking disabled) bit indicates in a query that Pending (non-authenticated) data is acceptable to the resolver sending the query.

These bits are allocated from the previously must-be-zero Z field as follows:



These bits are zero in old servers and resolvers. Thus the responses of old servers are not flagged as authenticated to security aware

resolvers and queries from non-security aware resolvers do not assert

the checking disabled bit and thus will be answered by security aware servers only with Authenticated or Insecure data. Security aware resolvers MUST NOT trust the AD bit unless they trust the server they are talking to and either have a secure path to it or use DNS transaction security.

Any security aware resolver willing to do cryptography SHOULD assert the CD bit on all queries to permit it to impose its own policies and to reduce DNS latency time by allowing security aware servers to answer with Pending data.

Security aware servers MUST NOT return Bad data. For non-security aware resolvers or security aware resolvers requesting service by having the CD bit clear, security aware servers MUST return only Authenticated or Insecure data in the answer and authority sections with the AD bit set in the response. Security aware servers SHOULD return Pending data, with the AD bit clear in the response, to security aware resolvers requesting this service by asserting the CD bit in their request. The AD bit MUST NOT be set on a response unless all of the RRs in the answer and authority sections of the response are either Authenticated or Insecure. The AD bit does not cover the additional information section.

6.2 Staticly Configured Keys

The public key to authenticate a zone SHOULD be defined in local configuration files before that zone is loaded at the primary server so the zone can be authenticated.

While it might seem logical for everyone to start with a public key associated with the root zone and staticly configure this in every resolver, this has problems. The logistics of updating every DNS resolver in the world should this key ever change would be severe. Furthermore, many organizations will explicitly wish their "interior" DNS implementations to completely trust only their own DNS servers. Interior resolvers of such organizations can then go through the organization's zone servers to access data outside the organization's domain and need not be configured with keys above the organization's DNS apex.

Host resolvers that are not part of a larger organization may be configured with a key for the domain of their local ISP whose recursive secure DNS caching server they use.

6.3 Chaining Through The DNS

Starting with one or more trusted keys for any zone, it should be possible to retrieve signed keys for that zone's subzones which have a key. A secure sub-zone is indicated by a KEY RR with non-null key information appearing with the NS RRs in the sub-zone and which may also be present in the parent. These make it possible to descend within the tree of zones.

6.3.1 Chaining Through KEYs

In general, some RRset that you wish to validate in the secure DNS will be signed by one or more SIG RRs. Each of these SIG RRs has a signer under whose name is stored the public KEY to use in authenticating the SIG. Each of those KEYs will, generally, also be signed with a SIG. And those SIGs will have signer names also referring to KEYs. And so on. As a result, authentication leads to chains of alternating SIG and KEY RRs with the first SIG signing the original data whose authenticity is to be shown and the final KEY being some trusted key staticly configured at the resolver performing the authentication.

In testing such a chain, the validity periods of the SIGs encountered must be intersected to determine the validity period of the authentication of the data, a purely algorithmic process. In addition, the validation of each SIG over the data with reference to a KEY must meet the objective cryptographic test implied by the cryptographic algorithm used (although even here the resolver may have policies as to trusted algorithms and key lengths). Finally, the judgement that a SIG with a particular signer name can authenticate data (possibly a KEY RRset) with a particular owner name, is primarily a policy question. Ultimately, this is a policy local to the resolver and any clients that depend on that resolver's decisions. It is, however, recommended, that the policy below be adopted:

> Let A < B mean that A is a shorter domain name than B formed by dropping one or more whole labels from the left end of B, i.e., A is a direct or indirect superdomain of B. Let A = B mean that A and B are the same domain name (i.e., are identical after letter case canonicalization). Let A > B mean that A is a longer domain name than B formed by adding one or more whole labels on the left end of B, i.e., A is a direct or indirect subdomain of B
>
> Let Static be the owner names of the set of staticly configured trusted keys at a resolver.

Then Signer is a valid signer name for a SIG authenticating an RRset (possibly a KEY RRset) with owner name Owner at the resolver if any of the following three rules apply:

(1) Owner > or = Signer (except that if Signer is root, Owner must be root or a top level domain name). That is, Owner is the same as or a subdomain of Signer.

(2) (Owner < Signer) and (Signer > or = some Static). That is, Owner is a superdomain of Signer and Signer is staticly configured or a subdomain of a staticly configured key.

(3) Signer = some Static. That is, the signer is exactly some staticly configured key.

Rule 1 is the rule for descending the DNS tree and includes a special prohibition on the root zone key due to the restriction that the root zone be only one label deep. This is the most fundamental rule.

Rule 2 is the rule for ascending the DNS tree from one or more staticly configured keys. Rule 2 has no effect if only root zone keys are staticly configured.

Rule 3 is a rule permitting direct cross certification. Rule 3 has no effect if only root zone keys are staticly configured.

Great care should be taken that the consequences have been fully considered before making any local policy adjustments to these rules (other than dispensing with rules 2 and 3 if only root zone keys are staticly configured).

6.3.2 Conflicting Data

It is possible that there will be multiple SIG-KEY chains that appear to authenticate conflicting RRset answers to the same query. A resolver should choose only the most reliable answer to return and discard other data. This choice of most reliable is a matter of local policy which could take into account differing trust in algorithms, key sizes, staticly configured keys, zones traversed, etc. The technique given below is recommended for taking into account SIG-KEY chain length.

A resolver should keep track of the number of successive secure zones traversed from a staticly configured key starting point to any secure zone it can reach. In general, the lower such a distance number is, the greater the confidence in the data. Staticly configured data should be given a distance number of zero. If a query encounters

different Authenticated data for the same query with different

distance values, that with a larger value should be ignored unless some other local policy covers the case.

A security conscious resolver should completely refuse to step from a secure zone into a unsecured zone unless the unsecured zone is certified to be non-secure by the presence of an authenticated KEY RR for the unsecured zone with the no-key type value. Otherwise the resolver is getting bogus or spoofed data.

If legitimate unsecured zones are encountered in traversing the DNS tree, then no zone can be trusted as secure that can be reached only via information from such non-secure zones. Since the unsecured zone data could have been spoofed, the "secure" zone reached via it could be counterfeit. The "distance" to data in such zones or zones reached via such zones could be set to 256 or more as this exceeds the largest possible distance through secure zones in the DNS.

6.4 Secure Time

Coordinated interpretation of the time fields in SIG RRs requires that reasonably consistent time be available to the hosts implementing the DNS security extensions.

A variety of time synchronization protocols exist including the Network Time Protocol (NTP [RFC 1305, 2030]). If such protocols are used, they MUST be used securely so that time can not be spoofed. Otherwise, for example, a host could get its clock turned back and might then believe old SIG RRs, and the data they authenticate, which were valid but are no longer.

7. ASCII Representation of Security RRs

This section discusses the format for master file and other ASCII presentation of the three DNS security resource records.

The algorithm field in KEY and SIG RRs can be represented as either an unsigned integer or symbolicly. The following initial symbols are defined as indicated:

Value	Symbol
001	RSAMD5
002	DH
003	DSA
004	ECC
252	INDIRECT
253	PRIVATEDNS
254	PRIVATEOID

7.1 Presentation of KEY RRs

KEY RRs may appear as single logical lines in a zone data master file [RFC 1033].

The flag field is represented as an unsigned integer or a sequence of mnemonics as follows separated by instances of the verticle bar ("|") character:

BIT	Mnemonic	Explanation
0-1		key type
	NOCONF	=1 confidentiality use prohibited
	NOAUTH	=2 authentication use prohibited
	NOKEY	=3 no key present
2	FLAG2	- reserved
3	EXTEND	flags extension
4	FLAG4	- reserved
5	FLAG5	- reserved
6-7		name type
	USER	=0 (default, may be omitted)
	ZONE	=1
	HOST	=2 (host or other end entity)
	NTYP3	- reserved
8	FLAG8	- reserved
9	FLAG9	- reserved
10	FLAG10	- reserved
11	FLAG11	- reserved
12-15		signatory field, values 0 to 15
	can be represented by SIG0, SIG1, ... SIG15	

No flag mnemonic need be present if the bit or field it represents is zero.

The protocol octet can be represented as either an unsigned integer or symbolicly. The following initial symbols are defined:

```
000    NONE
001    TLS
002    EMAIL
003    DNSSEC
004    IPSEC
255    ALL
```

Note that if the type flags field has the NOKEY value, nothing appears after the algorithm octet.

The remaining public key portion is represented in base 64 (see Appendix A) and may be divided up into any number of white space separated substrings, down to single base 64 digits, which are concatenated to obtain the full signature. These substrings can span lines using the standard parenthesis.

Note that the public key may have internal sub-fields but these do not appear in the master file representation. For example, with algorithm 1 there is a public exponent size, then a public exponent, and then a modulus. With algorithm 254, there will be an OID size, an OID, and algorithm dependent information. But in both cases only a single logical base 64 string will appear in the master file.

7.2 Presentation of SIG RRs

A data SIG RR may be represented as a single logical line in a zone data file [RFC 1033] but there are some special considerations as described below. (It does not make sense to include a transaction or request authenticating SIG RR in a file as they are a transient authentication that covers data including an ephemeral transaction number and so must be calculated in real time.)

There is no particular problem with the signer, covered type, and times. The time fields appears in the form YYYYMMDDHHMMSS where YYYY is the year, the first 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), the second MM is the minute (00-59), and SS is the second (00-59).

The original TTL field appears as an unsigned integer.

If the original TTL, which applies to the type signed, is the same as the TTL of the SIG RR itself, it may be omitted. The date field which follows it is larger than the maximum possible TTL so there is no ambiguity.

The "labels" field appears as an unsigned integer.

The key tag appears as an unsigned number.

However, the signature itself can be very long. It is the last data field and is represented in base 64 (see Appendix A) and may be divided up into any number of white space separated substrings, down to single base 64 digits, which are concatenated to obtain the full signature. These substrings can be split between lines using the standard parenthesis.

7.3 Presentation of NXT RRs

NXT RRs do not appear in original unsigned zone master files since they should be derived from the zone as it is being signed. If a signed file with NXTs added is printed or NXTs are printed by debugging code, they appear as the next domain name followed by the RR type present bits as an unsigned interger or sequence of RR mnemonics.

8. Canonical Form and Order of Resource Records

This section specifies, for purposes of domain name system (DNS) security, the canonical form of resource records (RRs), their name order, and their overall order. A canonical name order is necessary to construct the NXT name chain. A canonical form and ordering within an RRset is necessary in consistently constructing and verifying SIG RRs. A canonical ordering of types within a name is required in connection with incremental transfer (Section 5.6.2).

8.1 Canonical RR Form

For purposes of DNS security, the canonical form for an RR is the wire format of the RR with domain names (1) fully expanded (no name compression via pointers), (2) all domain name letters set to lower case, (3) owner name wild cards in master file form (no substitution made for *), and (4) the original TTL substituted for the current TTL.

8.2 Canonical DNS Name Order

For purposes of DNS security, the canonical ordering of owner names is to sort individual labels as unsigned left justified octet strings where the absence of a octet sorts before a zero value octet and upper case letters are treated as lower case letters. Names in a

zone are sorted by sorting on the highest level label and then,

within those names with the same highest level label by the next lower label, etc. down to leaf node labels. Within a zone, the zone name itself always exists and all other names are the zone name with some prefix of lower level labels. Thus the zone name itself always sorts first.

Example:

```
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
```

8.3 Canonical RR Ordering Within An RRset

Within any particular owner name and type, RRs are sorted by RDATA as a left justified unsigned octet sequence where the absence of an octet sorts before the zero octet.

8.4 Canonical Ordering of RR Types

When RRs of the same name but different types must be ordered, they are ordered by type, considering the type to be an unsigned integer, except that SIG RRs are placed immediately after the type they cover. Thus, for example, an A record would be put before an MX record because A is type 1 and MX is type 15 but if both were signed, the order would be A < SIG(A) < MX < SIG(MX).

9. Conformance

Levels of server and resolver conformance are defined below.

9.1 Server Conformance

Two levels of server conformance for DNS security are defined as follows:

BASIC: Basic server compliance is the ability to store and retrieve

(including zone transfer) SIG, KEY, and NXT RRs. Any secondary or

caching server for a secure zone MUST have at least basic compliance and even then some things, such as secure CNAMEs, will not work without full compliance.

FULL: Full server compliance adds the following to basic compliance: (1) ability to read SIG, KEY, and NXT RRs in zone files and (2) ability, given a zone file and private key, to add appropriate SIG and NXT RRs, possibly via a separate application, (3) proper automatic inclusion of SIG, KEY, and NXT RRs in responses, (4) suppression of CNAME following on retrieval of the security type RRs, (5) recognize the CD query header bit and set the AD query header bit, as appropriate, and (6) proper handling of the two NXT RRs at delegation points. Primary servers for secure zones MUST be fully compliant and for complete secure operation, all secondary, caching, and other servers handling the zone SHOULD be fully compliant as well.

9.2 Resolver Conformance

Two levels of resolver compliance (including the resolver portion of a server) are defined for DNS Security:

BASIC: A basic compliance resolver can handle SIG, KEY, and NXT RRs when they are explicitly requested.

FULL: A fully compliant resolver (1) understands KEY, SIG, and NXT RRs including verification of SIGs at least for the mandatory algorithm, (2) maintains appropriate information in its local caches and database to indicate which RRs have been authenticated and to what extent they have been authenticated, (3) performs additional queries as necessary to attempt to obtain KEY, SIG, or NXT RRs when needed, (4) normally sets the CD query header bit on its queries.

10. Security Considerations

This document specifies extensions to the Domain Name System (DNS) protocol to provide data integrity and data origin authentication, public key distribution, and optional transaction and request security.

It should be noted that, at most, these extensions guarantee the validity of resource records, including KEY resource records, retrieved from the DNS. They do not magically solve other security problems. For example, using secure DNS you can have high confidence in the IP address you retrieve for a host name; however, this does

not stop someone for substituting an unauthorized host at that

address or capturing packets sent to that address and falsely responding with packets apparently from that address. Any reasonably complete security system will require the protection of many additional facets of the Internet beyond DNS.

The implementation of NXT RRs as described herein enables a resolver to determine all the names in a zone even if zone transfers are prohibited (section 5.6). This is an active area of work and may change.

A number of precautions in DNS implementation have evolved over the years to harden the insecure DNS against spoofing. These precautions should not be abandoned but should be considered to provide additional protection in case of key compromise in secure DNS.

11. IANA Considerations

KEY RR flag bits 2 and 8-11 and all flag extension field bits can be assigned by IETF consensus as defined in RFC 2434. The remaining values of the NAMTYP flag field and flag bits 4 and 5 (which could conceivably become an extension of the NAMTYP field) can only be assigned by an IETF Standards Action [RFC 2434].

Algorithm numbers 5 through 251 are available for assignment should sufficient reason arise. However, the designation of a new algorithm could have a major impact on interoperability and requires an IETF Standards Action [RFC 2434]. The existence of the private algorithm types 253 and 254 should satify most needs for private or proprietary algorithms.

Additional values of the Protocol Octet (5-254) can be assigned by IETF Consensus [RFC 2434].

The meaning of the first bit of the NXT RR "type bit map" being a one can only be assigned by a standards action.

References

[RFC 1033] - M. Lottor, "Domain Administrators Operations Guide", November 1987.

[RFC 1034] - P. Mockapetris, "Domain Names - Concepts and Facilities", STD 13, November 1987.

[RFC 1035] - P. Mockapetris, "Domain Names - Implementation and Specifications", STD 13, November 1987.

[RFC 1305] - D. Mills, "Network Time Protocol (v3)", March 1992.

[RFC 1530] - C. Malamud, and M. Rose, "Principles of Operation for the TPC.INT Subdomain: General Principles and Policy", October 1993.

[RFC 1825] - Ran Atkinson, "Security Architecture for the Internet Protocol", August 1995.

[RFC 1982] - Robert Elz, Rrandy Bush, "Serial Number Arithmetic", 09/03/1996.

[RFC 1995] - Masatka Ohta, "Incremental Zone Transfer in DNS", August 1996.

[RFC 2030] - D. Mills, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", October 1996.

[RFC 2045] - N. Freed & N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", November 1996.

[RFC 2065] - Donald Eastlake 3rd, Charles Kaufman, "Domain Name System Security Extensions", 01/03/1997.

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

[RFC 2136] - P. Vixie, S. Thomson, Y. Rekhter, J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", 04/21/1997.

[RFC 2137] - Donald Eastlake 3rd, "Secure Domain Name System Dynamic Update", 04/21/1997.

[RFC 2181] - Robert Elz, Randy Bush, "Clarifications to the DNS Specification", July 1997.

[RFC 2434] - T. Narten, H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", October 1998.

draft-ietf-dnssec-rsa-*.txt, D. Eastlake, "RSA/MD5 KEYs and SIGs in the Domain Name System (DNS)".

draft-ietf-dnssec-dhk-*.txt, D. Eastlake, "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)".

draft-ietf-dnssec-dss-*.txt, D. Eastlake, "DSA KEYs and SIGs in the Domain Name System (DNS)".

draft-ietf-dsnssec-certs-*.txt, D. Eastlake, O. Gudmundsson, "Storing Certificates in the Domain Name System".

draft-ietf-dnssec-secops-*.txt, D. Eastlake, "DNS Operational Security Considerations".

[RSA FAQ] - RSADSI Frequently Asked Questions periodic posting.

Author's Address

Donald E. Eastlake 3rd
IBM
65 Shindegan Hill Road
RR #1
Carmel, NY 10512

Telephone: +1-914-784-7913 (w)
+1-914-276-2668 (h)
fax: +1-914-784-3833 (w-fax)
email: dee3@us.ibm.com

Expiration and File Name

This draft expires June 1999.

Its file name is draft-ietf-dnssec-secext2-07.txt.

Appendix A: Base 64 Encoding

The following encoding technique is taken from [RFC 2045] by N. Borenstein and N. Freed. It is reproduced here in an edited form for convenience.

A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=", is used to signify a special processing function.)

The encoding process represents 24-bit groups of input bits as output strings of 4 encoded characters. Proceeding from left to right, a 24-bit input group is formed by concatenating 3 8-bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each of which is translated into a single digit in the base 64 alphabet.

Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string.

Table 1: The Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a quantity. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Padding at the end of the data is performed using the '=' character. Since all base 64 input is an integral number of octets, only the following cases

can arise: (1) the final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output will be

an integral multiple of 4 characters with no "=" padding, (2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or (3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character.

Appendix B: Changes from RFC 2065

This section summarizes the most important changes that have been made since RFC 2065.

1. Most of Section 7 of [RFC 2065] called "Operational Considerations", has been removed and may be made into a separate document [draft-ietf-dnssec-secops-*.txt].

2. The KEY RR has been changed by (2a) eliminating the "experimental" flag as unnecessary, (2b) reserving a flag bit for flags expansion, (2c) more compactly encoding a number of bit fields in such a way as to leave unchanged bits actually used by the limited code currently deployed, (2d) eliminating the IPSEC and email flag bits which are replaced by values of the protocol field and adding a protocol field value for DNS security itself, (2e) adding material to indicate that zone KEY RRs occur only at delegation points, and (2f) removing the description of the RSA/MD5 algorithm to a separate document [draft-ietf-dnssec-rsa-*.txt]. Section 3.4 describing the meaning of various combinations of "no-key" and key present KEY RRs has been added and the secure / unsecure status of a zone has been clarified as being per algorithm.

3. The SIG RR has been changed by (3a) renaming the "time signed" field to be the "signature inception" field, (3b) clarifying that signature expiration and inception use serial number ring arithmetic, (3c) changing the definition of the key footprint/tag for algorithms other than 1 and adding Appendix C to specify its calculation. In addition, the SIG covering type AXFR has been eliminated while one covering IXFR [RFC 1995] has been added (see section 5.6).

4. Algorithm 3, the DSA algorithm, is now designated as the mandatory to implement algorithm. Algorithm 1, the RSA/MD5 algorithm, is now a recommended option. Algorithm 2 and 4 are designated as the Diffie-Hellman key and elliptic cryptography algorithms respectively, all to be defined in separate documents. Algorithm code point 252 is designated to indicate "indirect" keys, to be defined in a separate document, where the actual key is elsewhere. Both the KEY and SIG RR definitions have been simplified by eliminating the "null" algorithm 253 as defined in [RFC 2065]. That algorithm had been included because at the time it was thought it might be useful in DNS dynamic update [RFC 2136]. It was in fact not so used and it is dropped to simplify DNS security. Howver, that algorithm number has been re-used to indicate private algorithms where a domain name specifies the algorithm.

5. The NXT RR has been changed so that (5a) the NXT RRs in a zone cover all names, including wildcards as literal names without

expansion, except for glue address records whose names would not otherwise appear, (5b) all NXT bit map areas whose first octet has bit zero set have been reserved for future definition, (5c) the number of and circumstances under which an NXT must be returned in connection with wildcard names has been extended, and (5d) in connection with the bit map, references to the WKS RR have been removed and verticle bars ("|") have been added between the RR type mnemonics in the ASCII representation.

6. Information on the canonical form and ordering of RRs has been moved into a separate Section 8.

7. A subsection covering incremental and full zone transfer has been added in Section 5.

8. Concerning DNS chaining: Further specification and policy recommendations on secure resolution have been added, primarily in Section 6.3.1. It is now clearly stated that authenticated data has a validity period of the intersection of the validity periods of the SIG RRs in its authentication chain. The requirement to staticly configure a superzone's key signed by a zone in all of the zone's authoritative servers has been removed. The recommendation to continue DNS security checks in a secure island of DNS data that is separated from other parts of the DNS tree by insecure zones and does not contain a zone for which a key has been staticly configured was dropped.

9. It was clarified that the presence of the AD bit in a response does not apply to the additional information section or to glue address or delegation point NS RRs. The AD bit only indicates that the answer and authority sections of the response are authoritative.

10. It is now required that KEY RRs and NXT RRs be signed only with zone-level keys.

11. Add IANA Considerations section and references to RFC 2434.

Appendix C: 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, is in ANSI C. It is coded for clarity, not efficiency. (See section 4.1.6 for how to determine the Key Tag of an algorithm 1 key.)

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