Network Working Group
Request for Comments: 2065
Updates: 1034, 1035
Category: Standards Track

D. Eastlake, 3rd
CyberCash
C. Kaufman
Iris

January 1997

Domain Name System Security Extensions

Status of this Memo

Abstract

The Domain Name System (DNS) has become a critical operational part of the Internet infrastructure yet it has no strong security mechanisms to assure data integrity or authentication. Extensions to the DNS are described that provide these services to security aware resolvers or applications through the use of cryptographic digital signatures. These digital signatures are included in secured zones as resource records. Security can still be provided even through non-security aware DNS servers in many cases.

The extensions also provide for the storage of authenticated public keys in the DNS. This storage of keys can support general public key distribution service 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.

Acknowledgments

The significant contributions of the following persons (in alphabetic order) to this document are gratefully acknowledged:

Harald T. Alvestrand
Madelyn Badger
Scott Bradner
Matt Crawford
James M. Galvin
Olafur Gudmundsson
Edie Gunter
Sandy Murphy
Masataka Ohta
Michael A. Patton
Jeffrey I. Schiller

Table of Contents

1. Overview of Contents

This document describes 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, and 1035.

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, use in DNS responses, and file representation. 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, use in DNS responses, and file representation. 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 and its use in DNS responses. The NXT RR permits authenticated denial in the DNS of the existence of a name or of a particular 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 all combinations of security aware and security non-aware. Two additional query header bits are defined for signaling between resolvers and servers.

Section 7 reviews a variety of operational considerations including key generation, lifetime, and storage.

Section 8 defines levels of conformance for resolvers and servers.

Section 9 provides a few paragraphs on overall security considerations.

An Appendix is provided that gives details of base 64 encoding which is used in the file representation of some RR's defined in this document.

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.

In addition, no effort has been made to provide for any confidentiality for queries or responses. (This service may be available via IPSEC [RFC 1825].)

2.2 Key Distribution

Resource records (RRs) are defined to associate keys with DNS names. This permits the DNS to be used as a public key distribution mechanism in support of the DNS data origin authentication and other security services.

The syntax of a KEY resource record (RR) is described in Section 3. It includes an algorithm identifier, the actual public key parameters, 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 3.7, 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 records in the DNS cryptographically generated digital signatures. Commonly, there will be a single private key that signs for an entire zone. If a security aware resolver reliably learns the public key of the zone, it can verify, for signed data read from that zone, that it was properly authorized and is reasonably current. The expected implementation is for the zone private key to be kept off-line and used to re-sign all of the records in the zone periodically.

This data origin authentication key belongs to the zone and not to the servers that store copies of the data. That means compromise of a server or even all servers 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 can learn the public key of a zone either by reading it from DNS or by having it staticly configured. To reliably learn the public key by reading it from DNS, the key itself must be signed. Thus, to provide a reasonable degree of security, the resolver must be configured with at least the public key of one zone that it can use to authenticate signatures. From there, it can securely read the public keys of other zones, if the intervening zones in the DNS tree are secure and their signed keys accessible. (It is in principle more secure to have the resolver manually configured with the public

keys of multiple zones, since then the compromise of a single zone would not permit the faking of information from other zones. It is also more administratively cumbersome, however, particularly when public keys change.)

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, as a practical matter, the key resource type needed for key distribution. This service can be supported by existing resolver and server implementations so long as they can support the additional resource types (see Section 8). The one exception is that CNAME referrals from a secure zone can not be authenticated if they are from non-security aware servers (see Section 2.3.5).

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

2.3.1 The SIG Resource Record

The syntax of a SIG resource record (signature) is described in Section 4. It includes the type of the RR(s) being signed, the name of the signer, the time at which the signature was created, the time it expires (when it is no longer to be believed), its original time to live (which may be longer than its current time to live but cannot be shorter), the cryptographic algorithm in use, and the actual signature.

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 RRs and delgation point NS RRs. A security aware server supporting the performance enhanced version of the DNS protocol security extensions will attempt to return, with RRs retrieved, the corresponding SIGs. If a server does not support the protocol, the resolver must retrieve all the SIG records for a name and select the one or ones that sign the resource record(s) that resolver is interested in.

2.3.2 Authenticating Name and Type Non-existence

The above security mechanism provides only a way to sign existing RRs 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 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 field tick down when resource records 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 time to live values undetected. Instead, we include the "original" time-to-live in the signature and communicate that data in addition to the current time-to-live. Unscrupulous servers under this scheme can manipulate the time to live but a security aware resolver will bound the TTL value it uses at the original signed value. Separately, signatures include a time signed and an expiration time. A resolver that knows the absolute time can determine securely whether a signature has expired. 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, in any case, 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 and signed by the zone’s key. But the operational DNS 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 may 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.

In general, there must be a zone KEY RR for the subzone in the superzone and the copy signed in the superzone is controlling. For all but one other RR type that should appearing in both the superzone

and subzone, the data from the subzone is more authoritative. To avoid conflicts, only the KEY RR in the superzone should be signed and the NS and any A (glue) 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 there. The NXT RR type is an 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 RRs

There is a significant 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 will not retrieve any associated signature, key, or NXT RR. For types other than CNAME, it will retrieve that type at the target name of the CNAME (or chain of CNAMEs) and will return the CNAME as additional information. 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.

In general, 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 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 two cases where a SIG resource record is signed by other than the zone private key. One is for support of dynamic update where an entity is permitted to authenticate/update its own records. The public key of the entity must be present in the DNS and be appropriately signed but the other RR(s) may be signed with the entity’s key. The other is for support of transaction and request authentication as described in Section 2.4 immediately below.

2.4 DNS Transaction and Request Authentication

The data origin authentication service described above protects retrieved resource records but provides no protection for DNS requests or for message headers.

If header bits are falsely set by a 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, that the response is from the query it sent, and 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 the current DNS and requests with a non-empty additional information section are ignored by almost all current DNS servers. However, this syntax for signing requests is defined in connection with authenticating future secure dynamic update requests or the like.

The private keys used in transaction and request security belongs to the host composing the request or reply message, not to the zone involved. The corresponding public key is normally stored in and retrieved from the DNS.

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.

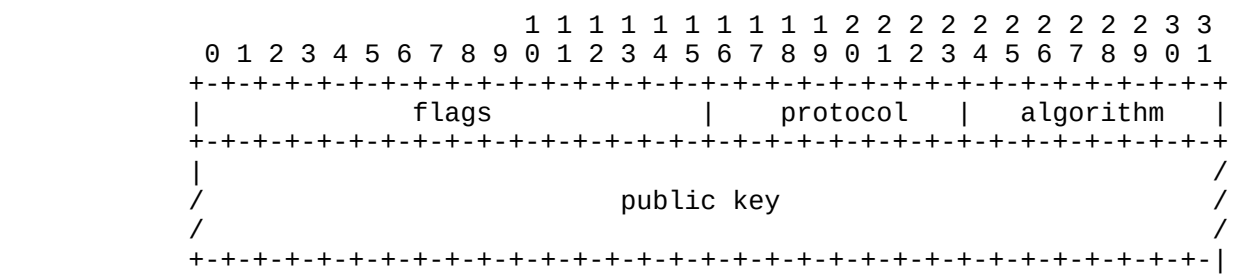
3. The KEY Resource Record

The KEY resource record (RR) is used to document a key that is associated with a Domain Name System (DNS) name. It will be a public key as only public keys are stored in the DNS. This can be the public key of a zone, a host or other end entity, or a user. A KEY RR is, like any other RR, authenticated by a SIG RR. Security aware DNS implementations MUST be designed to handle at least two simultaneously valid keys of the same type associated with a name.

The type number for the KEY RR is 25.

3.1 KEY RDATA format

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



The meaning of the KEY RR owner name, flags, and protocol octet are described in Sections 3.2, 3.3 and 3.4 below respectively. The flags and algorithm must be examined before any data following the algorithm octet as they control the format and even whether there is any following data. The algorithm and public key fields are described in Section 3.5. The format of the public key is algorithm dependent.

3.2 Object Types, DNS Names, and Keys

The public key in a KEY RR belongs to the object named in the owner name.

This DNS name may refer to up to three different categories of things. For example, dee.cybercash.com could be (1) a zone, (2) a host or other end entity , and (3) the mapping into a DNS name of the user or account dee@cybercash.com. Thus, there are flags, 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 at every leaf node which is a delegation point (and thus the same owner name as the apex of a subzone) within a secure zone.

Although the same name can be used for up to all three of these categories, such overloading of a name is discouraged. It is also possible to use the same key for different things with the same name or even different names, but this is strongly discouraged. In particular, the use of a zone key as a non-zone key will usually require that the corresponding private key be kept on line and thereby become more vulnerable.

In addition to the name type bits, there are additional flag bits including the "type" field, "experimental" bit, "signatory" field, etc., as described below.

3.3 The KEY RR Flag Field

In the "flags" field:

Bit 0 and 1 are the key "type" field. Bit 0 a one indicates that use of the key is prohibited for authentication. Bit 1 a one indicates that use of the key is prohibited for confidentiality. If this field is zero, then 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. If both bits of this field 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.

Bit 2 is the "experimental" bit. It is ignored if the type field indicates "no key" and the following description assumes that type field to be non-zero. Keys may be associated with zones, entities, or users for experimental, trial, or optional use, in which case this bit will be one. If this bit is a zero, it means that the use or availability of security based on the key is "mandatory". Thus, if this bit is off for a zone key, the zone should be assumed secured by SIG RRs and any responses indicating the zone is not secured should be considered bogus. If this bit is a one for a host or end entity, it might sometimes operate in a secure mode and at other times operate without security. The experimental bit, like all other aspects of the KEY RR, is only effective if the KEY RR is appropriately signed by a SIG RR. The experimental bit must be zero for safe secure operation and should only be a one for a minimal transition period.

Bits 3-4 are reserved and must be zero.

Bit 5 on 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.domain could have a public key associated through a KEY RR with name j\.random_user.host.subdomain.domain and the user bit a one. It could be used in an 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.

Bit 6 on 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]. This is the public key used in connection with the optional DNS transaction authentication service if the owner name is a DNS server host. It could also be used in an IP-security protocol where authentication of at the host, rather than user, level was desired, such as routing, NTP, etc.

Bit 7 is the "zone" bit and 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 DNS data origin authentication.

Bit 8 is reserved to be the IPSEC [RFC 1825] bit 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 bits are on. The presence of a KEY resource with the IPSEC and entity bits on and experimental and no-key bits off is an assertion that the host speaks IPSEC.

Bit 9 is reserved to be the "email" bit and indicate that this key is valid for use in conjunction with MIME security multiparts. 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 bits are on.

Bits 10-11 are reserved and must be zero.

Bits 12-15 are the "signatory" field. If non-zero, they indicate that the key can validly sign RRs or updates of the same name. If the owner name is a wildcard, then RRs or updates with any name which is in the wildcard’s scope can be signed. Fifteen different non-zero values are possible for this field and any differences in their meaning are reserved for definition in connection with DNS dynamic update or other new DNS commands. Zone keys always have authority to sign any RRs in the zone regardless of the value of this field. The signatory field, like all other aspects of the KEY RR, is only effective if the KEY RR is appropriately signed by a SIG RR.

3.4 The Protocol Octet

It is anticipated that some keys stored in DNS will be used in conjunction with Internet protocols other than DNS (keys with zone bit or signatory field non-zero) and IPSEC/email (keys with IPSEC and/or email bit set). The protocol octet is provided to indicate that a key is valid for such use and, for end entity keys or the host part of user keys, that the secure version of that protocol is implemented on that entity or host.

Values between 1 and 191 decimal inclusive are available for assignment by IANA for such protocols. The 63 values between 192 and 254 inclusive will not be assigned to a specific protocol and are available for experimental use under bilateral agreement. Value 0 indicates, for a particular key, that it is not valid for any particular additional protocol beyond those indicated in the flag field. And value 255 indicates that the key is valid for all assigned protocols (those in the 1 to 191 range).

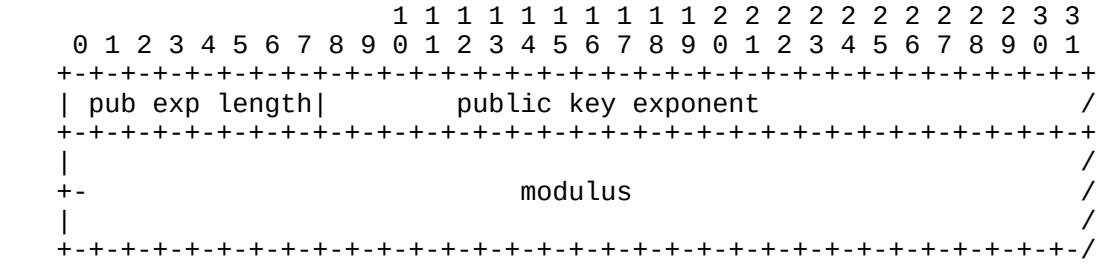
It is intended that new uses of DNS stored keys would initially be implemented, and operational experience gained, using the experimental range of the protocol octet. If demand for widespread deployment for the indefinite future warrants, a value in the assigned range would then be designated for the protocol. Finally, (1) should the protocol become so widespread in conjunction with other protocols and with which it shares key values that duplicate RRs are a serious burden and (2) should the protocol provide substantial facilities not available in any protocol for which a flags field bit has been allocated, then one of the remaining flag field bits may be allocated to the protocol. When such a bit has been allocated, a key can be simultaneously indicated as valid for that protocol and the entity or host can be simultaneously flagged as implementing the secure version of that protocol, along with other protocols for which flag field bits have been assigned.

3.5 The KEY Algorithm Number and the MD5/RSA Algorithm

This octet is the key algorithm parallel to the same field for the SIG resource. The MD5/RSA algorithm described in this document is number 1. Numbers 2 through 252 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. Number 254 is reserved for private use and will never be assigned a specific algorithm. For number 254, the public key area shown in the packet diagram above will actually begin with a length byte followed by an Object Identifier (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. Number 253 is

reserved as the "expiration date algorithm" for use where the expiration date or other labeling fields of SIGs are desired without any actual security. It is anticipated that this algorithm will only be used in connection with some modes of DNS dynamic update. For number 253, the public key area is null. Values 0 and 255 are reserved.

If the type field does not have the "no key" value and the algorithm field is 1, indicating the MD5/RSA algorithm, the public key field is structured as follows:



To promote interoperability, the exponent and modulus are each limited to 2552 bits in length. The public key exponent is a variable length unsigned integer. Its length in octets is represented as one octet if it is in the range of 1 to 255 and by a zero octet followed by a two octet unsigned length if it is longer than 255 bytes. The public key modulus field is a multiprecision unsigned integer. The length of the modulus can be determined from the RDLENGTH and the preceding RDATA fields including the exponent. Leading zero bytes are prohibited in the exponent and modulus.

3.6 Interaction of Flags, Algorithm, and Protocol Bytes

Various combinations of the no-key type value, algorithm byte, protocol byte, and any protocol indicating flags (such as the reserved IPSEC flag) are possible. (Note that the zone flag bit being on or the signatory field being non-zero is effectively a DNS protocol flag on.) The meaning of these combinations is indicated below:

NK = no key type value
AL = algorithm byte
PR = protocols indicated by protocol byte or protocol 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.
0	x	0	Illegal, claims key but has bad algorithm field.
0	x	1	Specified protocols insecure, others may be secure.
x	0	0	Useless. Gives key but no protocols to use it.
x	0	1	Useless. Denies key but for no protocols.
x	x	0	Specifies key for protocols and asserts that those protocols are implemented with security.
x	x	1	Algorithm not understood for protocol.

(remember, in reference to the above table, that a protocol byte of 255 means all protocols with protocol byte values assigned)

3.7 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.

Security aware DNS servers MUST include KEY RRs as additional information in responses where appropriate including the following:

(1) On the retrieval of NS RRs, the zone key KEY RR(s) for the zone served by these name servers MUST be included as additional information if space is avilable. There will always be at least one such KEY RR in a secure zone, even if it has the no-key type value to indicate that the subzone is insecure. If not all additional information will fit, the KEY RR(s) have higher priority than type A or AAAA glue RRs. If such a KEY RR does not fit on a retrieval, the retrieval must be considered truncated.

(2) On retrieval of type A or AAAA RRs, the end entity KEY RR(s) MUST be included if space is available. On inclusion of A or AAAA RRs as additional information, their KEY RRs will also be included but with lower priority than the relevant A or AAAA RRs.

3.8 File Representation of KEY RRs

KEY RRs may appear as lines in a zone data master file.

The flag field, protocol, and algorithm number octets are then represented as unsigned integers. Note that if the type field has the "no key" value or the algorithm specified is 253, nothing appears after the algorithm octet.

The remaining public key portion is represented in base 64 (see Appendix) 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.

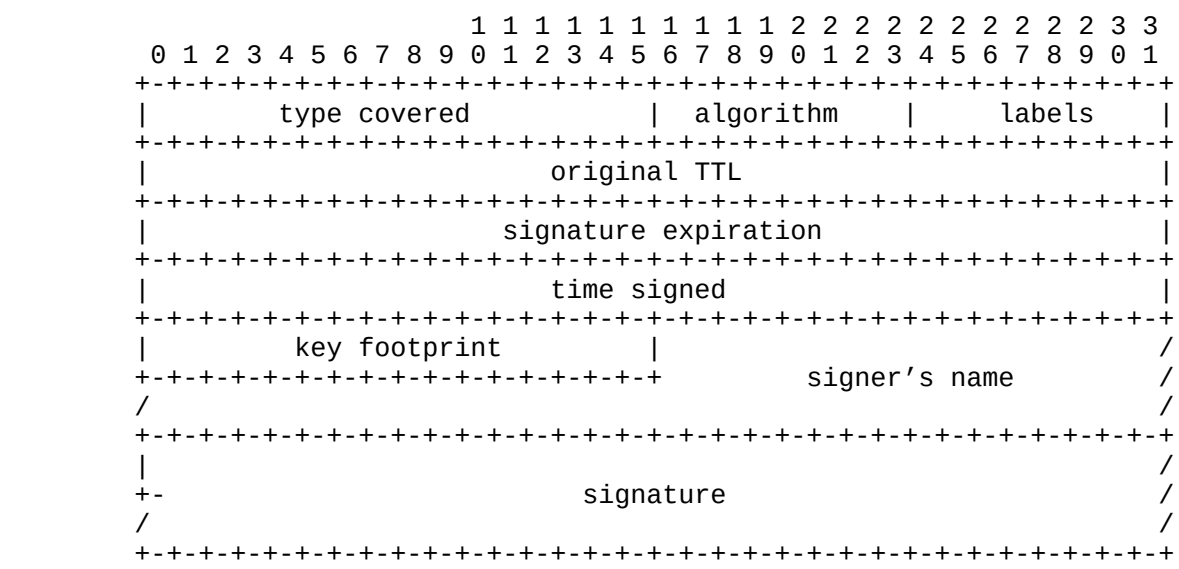
4. The SIG Resource Record

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 other RRs of a particular type, class, and name and binds them 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.

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.



The value of the SIG RR type is 24.

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

The algorithm number is an octet specifying the digital signature algorithm used parallel to the algorithm octet for the KEY RR. The MD5/RSA algorithm described in this document is number 1. Numbers 2 through 252 are available for assignment should sufficient reason arise to allocate them. However, the designation of a new algorithm could have a major impact on the interoperability of the global DNS system and requires an IETF standards action. Number 254 is reserved for private use and will not be assigned a specific algorithm. For number 254, the "signature" area shown above will actually begin with a length byte followed by an Object Identifier (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. Number 253, known as the "expiration date algorithm", is used when the expiration date or other non-signature fields of the SIG are desired without any actual security. It is anticipated that this algorithm will only be used in connection with some modes of DNS dynamic update. For number 253, the signature field will be null. Values 0 and 255 are reserved.

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 helps optimize the determination of the original form thus reducing the effort in authenticating signed data.

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.

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. This implies that all RRs for a particular type, name, and class must have the same TTL to start with.

The SIG is valid until the "signature expiration" time which is an unsigned number of seconds since the start of 1 January 1970, GMT, ignoring leap seconds. (See also Section 4.4.) SIG RRs should not have a date signed significantly in the future. To prevent misordering of network requests to update a zone dynamically, monotonically increasing "time signed" dates may be necessary.

The "time signed" field is an unsigned number of seconds since the start of 1 January 1970, GMT, ignoring leap seconds.

A SIG RR with an expiration date before the time signed must be considered corrupt and ignored.

The "key footprint" is a 16 bit quantity that is used to help efficiently select between multiple keys which may be applicable and as a quick check that a public key about to be used for the computationally expensive effort to check the signature is possibly valid. Its exact meaning is algorithm dependent. For the MD5/RSA algorithm, 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 lest significant 24 bits of the modulus in network order.

The "signer's name" field is the domain name of the signer generating the SIG RR. This is the owner of the public KEY RR that can be used to verify the signature. It is frequently the zone which contained the RR(s) being authenticated. The signer's name may be compressed with standard DNS name compression when being transmitted over the network.

The structure of the "signature" field is described below.

4.1.1 Signature Data

Except for algorithm number 253 where it is null, the actual signature portion of the SIG RR binds the other RDATA fields to all of the "type covered" RRs with that owner name and class. These covered RRs are thereby authenticated. To accomplish this, a data sequence is constructed as follows:

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

where "|" is concatenation, RDATA is all the RDATA fields in the SIG RR itself before and not including the signature, and RR(s) are all the RR(s) of the type covered with the same owner name and class as the SIG RR in canonical form and order. How this data sequence is processed into the signature is algorithm dependent.

For purposes of DNS security, the canonical form for an RR is the RR with domain names (1) fully expanded (no name compression via pointers), (2) all domain name letters set to lower case, and (3) the original TTL substituted for the current TTL.

For purposes of DNS security, the canonical order for RRs is to sort them in ascending order by name, considering labels as a left justified unsigned octet sequence in network (transmission) order where a missing octet sorts before a zero octet. (See also ordering discussion in Section 5.1.) Within any particular name they are similarly sorted by type and then RDATA as a left justified unsigned octet sequence. EXCEPT that the type SIG RR(s) covering any particular type appear immediately after the other RRs of that type. (This special consideration for SIG RR(s) in ordering really only applies to calculating the AXFR SIG RR as explained in section 4.1.3 below.) Thus if at name a.b there are two A RRs and one KEY RR, their order with SIGs for concatenating the "data" to be signed would be as follows:

```
a.b.  A ....
a.b.  A ....
a.b.  SIG A ...
a.b.  KEY ...
a.b.  SIG KEY ...
```

SIGs covering type ANY should not be included in a zone.

4.1.2 MD5/RSA Algorithm Signature Calculation

For the MD5/RSA algorithm, the signature is as follows

```
hash = MD5 ( data )

signature = ( 01 | FF* | 00 | prefix | hash ) ** e (mod n)
```

where MD5 is the message digest algorithm documented in RFC 1321, "|" is concatenation, "e" is the private key exponent of the signer, and "n" is the modulus of the signer's public key. 01, FF, and 00 are fixed octets of the corresponding hexadecimal value. "prefix" is the ASN.1 BER MD5 algorithm designator prefix specified in PKCS1, that is,

```
hex 3020300c06082a864886f70d020505000410 [NETSEC].
```

This prefix is included to make it easier to use RSAREF or similar packages. The FF octet is repeated the maximum number of times such that the value of the quantity being exponentiated is one octet shorter than the value of n.

(The above specifications are identical to the corresponding part of Public Key Cryptographic Standard #1 [PKCS1].)

The size of n, including most and least significant bits (which will be 1) SHALL be not less than 512 bits and not more than 2552 bits. n and e SHOULD be chosen such that the public exponent is small.

Leading zeros bytes are not permitted in the MD5/RSA algorithm signature.

A public exponent of 3 minimizes the effort needed to decode a signature. Use of 3 as the public exponent may be weak for confidentiality uses since, if the same data can be collected encrypted under three different keys with an exponent of 3 then, using the Chinese Remainder Theorem, the original plain text can be easily recovered. This weakness is not significant for DNS because we seek only authentication, not confidentiality.

4.1.3 Zone Transfer (AXFR) SIG

The above SIG mechanisms assure the authentication of all zone signed RRs of a particular name, class and type. However, to efficiently assure the completeness and security of zone transfers, a SIG RR owned by the zone name must be created with a type covered of AXFR that covers all zone signed RRs in the zone and their zone SIGs but not the SIG AXFR itself. The RRs are ordered and concatenated for hashing as described in Section 4.1.1. (See also ordering discussion in Section 5.1.)

The AXFR SIG must be calculated last of all zone key signed SIGs in the zone. In effect, when signing the zone, you order, as described above, all RRs to be signed by the zone, and all associated glue RRs and delegation point NS RRs. You can then make one pass inserting all the zone SIGs. As you proceed you hash RRs to be signed into both an RRset hash and the zone hash. When the name or type changes you calculate and insert the RRset zone SIG, clear the RRset hash, and hash that SIG into the zone hash (note that glue RRs and delegation point NSs are not zone signed but zone apex NSs are). When you have finished processing all the starting RRs as described above, you can then use the cumulative zone hash RR to calculate and insert an AXFR SIG covering the zone. Of course any computational technique producing the same results as above is permitted.

The AXFR 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 AXFR SIG is otherwise meaningless. The AXFR SIG is only retrieved as part of a zone transfer. After validation of the AXFR SIG, the zone MAY be considered valid without verification of the internal zone signed SIGs in the zone; however, any RRs authenticated by SIGs signed by entity keys or the like MUST still be validated. The AXFR SIG SHOULD be transmitted first in a zone transfer so the

receiver can tell immediately that they may be able to avoid verifying other zone signed SIGs.

RRs which are authenticated by a dynamic update key and not by the zone key (see Section 3.2) are not included in the AXFR SIG. They may originate in the network and might not, in general, be migrated to the recommended off line zone signing procedure (see Section 7.2). Thus, such RRs are not directly signed by the zone, are not included in the AXFR SIG, and are protected against omission from zone transfers only to the extent that the server and communication can be trusted.

4.1.4 Transaction and Request SIGs

A response message from a security aware server may optionally contain a special SIG as the last item in 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 4.1.2) of the entire preceding DNS reply message, including DNS header but not the IP header, concatenated with the entire DNS query message that produced this response, including the query's DNS header but not its IP header. That is

```
data = full response (less final 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 at the begining or any preceding request SIGs at the end. Such request SIGs are included in the "data" used to form any optional response transaction SIG.

WARNING: Request SIGs are unnecessary for currently defined queries and will cause almost all existing DNS servers to completely ignore a query. However, such SIGs may be needed to authenticate future DNS secure dynamic update or other requests.

4.2 SIG RRs in the Construction of Responses

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

1. when an RR set is placed in a response, its SIG RR has a higher priority for inclusion than other 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 its SIG RR.

3. SIGs to authenticate non-authoritative data (glue records and NS RRs for subzones) are unnecessary and MUST NOT be sent. (Note that KEYs for subzones are controlling in a superzone so the superzone's signature on the KEY MUST be included (unless the KEY was additional information and the SIG did not fit).)

4. If a SIG covers any RR that would be in the answer section of the response, its automatic inclusion MUST be 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 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 4.1.4). Such SIG RRs are signed by the DNS server originating the response. Although the signer field MUST be the 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 higher priority for inclusion than any other RR in the response.

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 what it believes to be 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 an insecure server. (As explained in 4.2 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 reasonable 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 a set of RRs are invalid, then the set of RR(s) 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 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 authenticate RRs. If a resolver does not implement transaction and/or request SIGs, it MUST ignore them without error.

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

4.4 Signature Expiration, TTLs, and Validity

Security aware servers must not consider SIG RRs to authenticate anything after their expiration time and not consider any RR to be authenticated after its signatures have expired. Within that

constraint, 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. 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 signature expiration time. Thus, in general, the TTL on an transmitted RR would be

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

4.5 File Representation of SIG RRs

A SIG RR can be represented as a single logical line in a zone data file [RFC1033] 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 and algorithm fields appear as unsigned integers.

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 does not appear in the file representation as it can be calculated from the owner name.

The key footprint appears as an unsigned decimal number.

However, the signature itself can be very long. It is the last data field and is represented in base 64 (see Appendix) 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.

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 is it not clear above how to authenticatably 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. A NXT RR and its SIG 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. This is a change in the existing standard 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 always result in an reply containing at least one signed RR.

NXT RRs do not appear in zone master files since they can be derived from the rest of the zone.

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 zone signed 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. The presence of the NXT RR means that generally no name between its owner name and the name in its RDATA area exists and that no other zone signed types exist under its owner name. This implies a canonical ordering of all domain names in a zone.

The ordering is to sort 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 are then 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. Since we are talking about 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.

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.

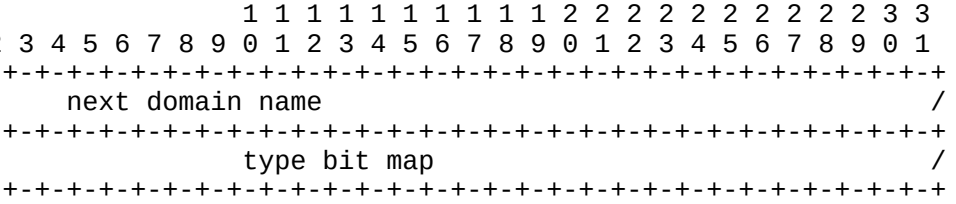
There are special considerations due to interaction with wildcards as explained below.

The NXT RRs for a zone SHOULD be automatically calculated and added to the zone by the same recommended off-line process that signs the zone (see Section 7.2). The NXT RR's TTL SHOULD not exceed the zone minimum TTL.

5.2 NXT RDATA Format

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

The type number for the NXT RR is 30.



The NXT RR type bit map is one bit per RR type present for the owner name similar to the WKS socket bit map. The first bit represents RR type zero (an illegal type which should not be present.) 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. The NXT bit map should be printed as a list of RR type mnemonics or decimal numbers similar to the WKS RR.

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 Example

Assume zone foo.tld has entries for

```
big.foo.tld,
medium.foo.tld.
small.foo.tld.
tiny.foo.tld.
```

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

```
big.foo.tld. NXT medium.foo.tld. A MX SIG NXT
big.foo.tld. SIG NXT 1 3 ( ;type-cov=NXT, alg=1, labels=3
                 19960102030405 ;signature expiration
                 19951211100908 ;time signed
                 21435          ;key footprint
                 foo.tld.       ;signer
 MxFcby9k/yvedMfQgKzhH5er0Mu/vILz45IkskceFGgiWCn/GxHhai6VAuHAoNUz4YoU
 1tVfSCSqQYn6//11U6Nld80jEeC8aTrO+KKmCaY= ;signature (640 bits)
                      )
```

Note that this response implies that big.foo.tld 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.tld, which is a non-existent name.

5.4 Interaction of NXT RRs and Wildcard RRs

Since, in some sense, a wildcard RR causes all possible names in an interval to exist, there should not be an NXT RR that would cover any part of this interval. Thus if *.X.ZONE exists you would expect an NXT RR that ends at X.ZONE and one that starts with the last name covered by *.X.ZONE. However, this "last name covered" is something very ugly and long like \255\255\255....X.zone. So the NXT for the interval following is simply given the owner name *.X.ZONE and an RDATA of the next name after the wildcard. This "*" type owner name is not expanded when the NXT is returned as authority information in connection with a query for a non-existent name.

If there could be any wildcard RRs in a zone and thus wildcard NXTs, care must be taken in interpreting the results of explicit NXT retrievals as the owner name may be a wildcard expansion.

The existence of one or more wildcard RRs covering a name interval makes it possible for a malicious server to hide any more specifically named RRs in the internal. The server can just falsely

return the wildcard match NXT instead of the more specifically named RRs. If there is a zone wide wildcard, there will be an NXT RR whose owner name is the wild card and whose RDATA is the zone name. In this case a server could conceal the existence of any more specific RRs in the zone. It would be possible to design a more strict NXT feature which would eliminate this possibility. But it would be more complex and might be so constraining as to make any dynamic update feature very difficult.

5.5 Blocking NXT Pseudo-Zone Transfers

In a secure zone, a resolver can query for the initial NXT associated with the zone name. Using the next domain name RDATA field from that RR, it can query for the next NXT RR. By repeating this, it can walk through all the NXTs in the zone. If there are no wildcards, it can use this technique to find all names in a zone. If it does type ANY queries, it can incrementally get all information in the zone and thus defeat attempts to administratively block zone transfers.

If there are any wildcards, this NXT walking technique will not find any more specific RR names in the part of the name space the wildcard covers. By doing explicit retrievals for wildcard names, a resolver could determine what intervals are covered by wildcards but still could not, with these techniques, find any names inside such intervals except by trying every name.

If it is desired to block NXT walking, the recommended method is to add a zone wide wildcard of the KEY type with the no-key type value and with no type (zone, entity, or user) bit on. This will cause there to be one zone covering NXT RR and leak no information about what real names exist in the zone. This protection from pseudo-zone transfers is bought at the expense of eliminating the data origin authentication of the non-existence of names that NXT RRs can provide. If an entire zone is covered by a wildcard, a malicious server can return an RR produced by matching the resulting wildcard NXT and can thus hide all the real data and delegations in the zone that have more specific names.

5.6 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 distinguished by their signers and next domain name fields. 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.

Insecure servers will never automatically return an NXT and some implementations may only return the NXT from the subzone on explicit queries.

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

Retrieving or resolving authentic data from the Domain Name System (DNS) involves starting with one or more trusted public keys for one or more zones. With trusted keys, a resolver willing to perform cryptography can progress securely through the secure DNS zone 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 a starting point.

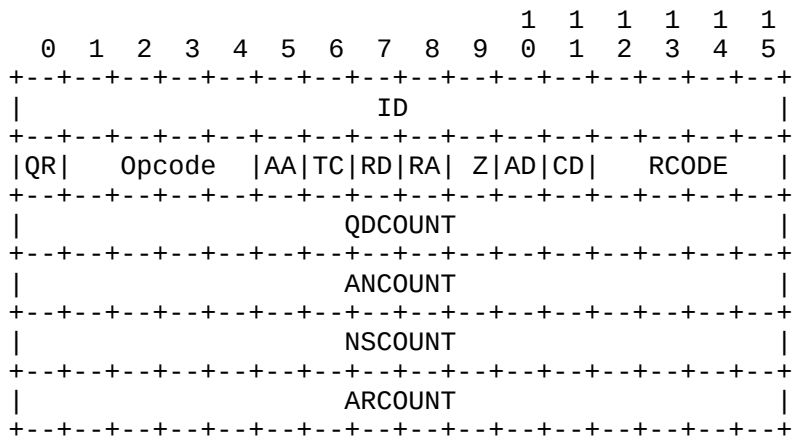
Data stored at a security aware server needs to be internally categorized as Authenticated, Pending, or Insecure. There is also a fourth transient state of Bad which indicates that all SIG checks have explicitly failed on the data. Such Bad data is not retained at a security aware server. Authenticated means that the data has a valid SIG under a KEY traceable via a chain of zero or more SIG and KEY RRs to a KEY configured at the resolver via its boot file. 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 because it is in or has been reached via a non-secured zone. 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 the data included has been verified by the server providing it. The CD (checking disabled) bit indicates in a query that non-verified data is acceptable to the resolver sending the query.

These bits are allocated from the 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 data. 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 reduce DNS latency time by allowing security aware servers to answer before they have resolved the validity of data.

Security aware servers NEVER 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 with the AD bit set in the response. Security aware resolvers will know that if data is Insecure versus Authentic by the absence of SIG RRs. Security aware servers MAY return Pending data to security aware resolvers requesting the service by clearing the AD bit in the response. The AD bit MUST NOT be set on a response unless all of the RRs in the response are either Authenticated or Insecure.

6.2 Boot File Format

Two boot file directives are added as described in this section.

The format for a boot file directive to configure a starting zone key is as follows:

```
pubkey name flags protocol algorithm key-data
```

for a public key. "name" is the owner name (if the line is translated into a KEY RR). Flags indicates the type of key and is the same as the flag octet in the KEY RR. Protocol and algorithm also have the same meaning as they do in the KEY RR. The material after the algorithm is algorithm dependent and, for private algorithms (algorithm 254), starts with the algorithm's identifying OID and its length. If the "no key" type value is set in flags or the algorithm is specified as 253, then the key-data after algorithm is null. When present the key-data is treated as an octet stream and encoded in base 64 (see Appendix).

A file of keys for cross certification or other purposes can be configured though the keyfile directive as follows:

```
keyfile filename
```

The file looks like a master file except that it can only contain KEY and SIG RRs with the SIGs signed under a key configured with the pubkey directive.

While it might seem logical for everyone to start with the key for the root zone, this has problems. The logistics of updating every DNS resolver in the world when the root key changes would be excessive. It may be some time before there even is a root key. Furthermore, many organizations will explicitly wish their "interior" DNS implementations to completely trust only their own zone. Such interior resolvers can then go through the organization's zone servers to access data outsize the organization's domain and should only be configured with the key forthe organization's DNS apex.

6.3 Chaining Through Zones

Starting with one or more trusted keys for a zone, it should be possible to retrieve signed keys for its subzones which have a key and, if the zone is not root, for its superzone. Every authoritative secure zone server MUST also include the KEY RR for a super-zone signed by the secure zone via a keyfile directive. This makes it possible to climb the tree of zones if one starts below root. A secure sub-zone is indicated by a KEY RR with non-null key

information appearing with the NS RRs for the sub-zone. These make it possible to descend within the tree of zones.

A resolver should keep track of the number of successive secure zones traversed from a 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. Data configured via a boot file directive should be given a distance number of zero. If a query encounters different data for the same query with different distance values, that with a larger value should be ignored.

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

If legitimate non-secure 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 non-secure zone data could have been spoofed, the "secure" zone reach via it could be counterfeit. The "distance" to data in such zones or zones reached via such zones could be set to 512 or more as this exceeds the largest possible distance through secure zones in the DNS. Nevertheless, continuing to apply secure checks within "secure" zones reached via non-secure zones is a good practice and will, as a practical matter, provide some small increase in security.

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, RFC1305). 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 and KEY RRs which were valid but no longer are.

7. Operational Considerations

This section discusses a variety of considerations in secure operation of the Domain Name System (DNS) using these protocol extensions.

7.1 Key Size Considerations

There are a number of factors that effect public key size choice for use in the DNS security extension. Unfortunately, these factors usually do not all point in the same direction. Choice of zone key size should generally be made by the zone administrator depending on their local conditions.

For most schemes, larger keys are more secure but slower. Given a small public exponent, verification (the most common operation) for the MD5/RSA algorithm will vary roughly with the square of the modulus length, signing will vary with the cube of the modulus length, and key generation (the least common operation) will vary with the fourth power of the modulus length. The current best algorithms for factoring a modulus and breaking RSA security vary roughly with the 1.6 power of the modulus itself. Thus going from a 640 bit modulus to a 1280 bit modulus only increases the verification time by a factor of 4 but increases the work factor of breaking the key by over 2^900. An upper bound of 2552 bits has been established for the MD5/RSA DNS security algorithm for interoperability purposes.

However, larger keys increase the size of the KEY and SIG RRs. This increases the chance of DNS UDP packet overflow and the possible necessity for using higher overhead TCP in responses.

The recommended minimum RSA algorithm modulus size, 640 bits, is believed by the authors to be secure at this time but high level zones in the DNS tree may wish to set a higher minimum, perhaps 1000 bits, for security reasons. (Since the United States National Security Agency generally permits export of encryption systems using an RSA modulus of up to 512 bits, use of that small a modulus, i.e. n, must be considered weak.)

For a key used only to secure data and not to secure other keys, 640 bits should be adequate at this time.

7.2 Key Storage

It is recommended that zone private keys and the zone file master copy be kept and used in off-line non-network connected physically secure machines only. Periodically an application can be run to add authentication to a zone by adding SIG and NXT RRs and adding no-key type KEY RRs for subzones where a real KEY RR is not provided. Then the augmented file can be transferred, perhaps by sneaker-net, to the networked zone primary server machine.

The idea is to have a one way information flow to the network to avoid the possibility of tampering from the network. Keeping the

zone master file on-line on the network and simply cycling it through an off-line signer does not do this. The on-line version could still be tampered with if the host it resides on is compromised. For maximum security, the master copy of the zone file should be off net and should not be updated based on an unsecured network mediated communication.

Note, however, that secure resolvers must be configured with some trusted on-line public key information (or a secure path to such a resolver) or they will be unable to authenticate.

Non-zone private keys, such as host or user keys, generally have to be kept on line to be used for real-time purposes such as DNS transaction security, IPSEC session set-up, or secure mail.

7.3 Key Generation

Careful key generation is a sometimes overlooked but absolutely essential element in any cryptographically secure system. The strongest algorithms used with the longest keys are still of no use if an adversary can guess enough to lower the size of the likely key space so that it can be exhaustively searched. Suggestions will be found in RFC 1750.

It is strongly recommended that key generation also occur off-line, perhaps on the machine used to sign zones (see Section 7.2).

7.4 Key Lifetimes

No key should be used forever. The longer a key is in use, the greater the probability that it will have been compromised through carelessness, accident, espionage, or cryptanalysis. Furthermore, if key rollover is a rare event, there is an increased risk that, when the time does come up change the key, no one at the site will remember how to do it or other problems will have developed in the procedures.

While key lifetime is a matter of local policy, these considerations suggest that no zone key should have a lifetime significantly over four years. A reasonable maximum lifetime for zone keys that are kept off-line and carefully guarded is 13 months with the intent that they be replaced every year. A reasonable maximum lifetime for end entity and useer keys that are used for IP-security or the like and are kept on line is 36 days with the intent that they be replaced monthly or more often. In some cases, an entity key lifetime of somewhat over a day may be reasonable.

7.5 Signature Lifetime

Signature expiration times must 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, if bad data or signatures were ever generated, mean a long time to flush such badness.

It is recommended that signature lifetime be a small multiple of the TTL but not less than a reasonable re-signing interval.

7.6 Root

It should be noted that in DNS the root is a zone unto itself. Thus the root zone key should only be seen signing itself or signing RRs with names one level below root, such as .aq, .edu, or .arpa. Implementations MAY reject as bogus any purported root signature of records with a name more than one level below root. The root zone contains the root KEY RR signed by a SIG RR under the root key itself.

8. Conformance

Levels of server and resolver conformance are defined.

8.1 Server Conformance

Two levels of server conformance are defined as follows:

> Minimal server compliance is the ability to store and retrieve (including zone transfer) SIG, KEY, and NXT RRs. Any secondary, caching, or other server for a secure zone MUST be at least minimally compliant and even then some things, such as secure CNAMEs, will not work without full compliance.

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 completely successful secure operation, all secondary, caching, and other servers handling the zone SHOULD be fully compliant as well.

8.2 Resolver Conformance

Two levels of resolver compliance are defined:

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

A fully compliant resolver (1) understands KEY, SIG, and NXT RRs, (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 from non-security aware servers, (4) normally sets the CD query header bit on its queries.

9. Security Considerations

This document describes technical details of extensions to the Domain Name System (DNS) protocol to provide data integrity and 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.

References

[NETSEC] - Network Security: PRIVATE Communications in a PUBLIC World, Charlie Kaufman, Radia Perlman, & Mike Speciner, Prentice Hall Series in Computer Networking and Distributed Communications 1995.

[PKCS1] - PKCS #1: RSA Encryption Standard, RSA Data Security, Inc., 3 June 1991, Version 1.4.

[RFC1032] - Stahl, M., "Domain Administrators Guide", RFC 1032, November 1987.

[RFC1033] - Lottor, M., "Domain Administrators Operations Guide", RRFC 1033, November 1987.

[RFC1034] - Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[RFC1035] - Mockapetris, P., "Domain Names - Implementation and Specifications", STD 13, RFC 1035, November 1987.

[RFC1305] - Mills, D., "Network Time Protocol (v3)", RFC 1305, March 1992.

[RFC1321] - Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC1530] - Malamud, C., and M. Rose, "Principles of Operation for the TPC.INT Subdomain: General Principles and Policy", RFC 1530, October 1993.

[RFC1750] - Eastlake, D., Crocker, S., and J, Schiller, "Randomness Requirements for Security", RFC 1750, December 1994.

[RFC1825] - Atkinson, R., "Security Architecture for the Internet Protocol", RFC 1825, August 1995.

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

Authors' Addresses

Donald E. Eastlake 3rd
CyberCash, Inc.
318 Acton Street
Carlisle, MA 01741 USA

Telephone: +1 508-287-4877
+1 508-371-7148(fax)
+1 703-620-4200(main office, Reston, Virginia, USA)
EMail: dee@cybercash.com

Charles W. Kaufman
Iris Associates
1 Technology Park Drive
Westford, MA 01886 USA

Telephone: +1 508-392-5276
EMail: charlie_kaufman@iris.com

Appendix: Base 64 Encoding

The following encoding technique is taken from RFC 1521 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.