DNSOP
Internet-Draft
Obsoletes: 2930 (if approved)
Intended status: Informational
Expires: 8 January 2026

D. Eastlake
Independent
M. Andrews
Internet Systems Consortium

7 July 2025

# Secret Key Agreement for DNS: The TKEY Resource Record
draft-eastlake-dnsop-rfc2930bis-tkey-03

## Abstract

RFC 8945 provides efficient authentication of Domain Name System (DNS) protocol messages using shared secret keys and the Transaction Signature (TSIG) resource record (RR). However, it provides no mechanism for setting up such keys other than by configuration. This document specifies the Transaction Key (TKEY) RR that can be used in a variety of modes to establish shared secret keys between a DNS resolver and server. This document obsoletes RFC 2930.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 January 2026.

## Copyright Notice

Table of Contents

## 1. Introduction

[[[NOTE: This draft contains occasional meta comments inside triple square brackets like this. These are intended to aid in the development of the document and would not appear in any resulting RFC.]]]

The Domain Name System (DNS) is a hierarchical, distributed, highly available database used for bi-directional mapping between domain names and IP addresses, for email routing, service access, and for other information [RFC1034] [RFC1035]. It has been extended to provide for public key-based data authentication [RFC4034] [RFC4035] and dynamic update [RFC3007]. Familiarity with these RFCs is assumed.

[RFC8945] provides efficient authentication of DNS messages using shared secret keys and the Transaction Signature (TSIG) resource record (RR) but provides no mechanism for setting up such keys other than by configuration. This document specifies the Transaction Key (TKEY) RR that can be used in a variety of modes to establish and delete such shared secret keys between a DNS resolver and server. This document obsoletes [RFC2930] and the significant changes from that RFC are list in Appendix B.

TKEY established keying materials and TSIGs that use them are associated with DNS servers and resolvers. They are not associated with DNS zones. They may be used to authenticate requests and transactions but they do not provide zone-based DNS RR data origin or denial of existence authentication [RFC4034] [RFC4035].

### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

General familiarity with DNS terminology [RFC9499] is assumed in this document. In all cases herein, the term "resolver" or "requester" includes that part of a server which may initiate requests including full and incremental [RFC1995] zone transfer queries, forwarded recursive queries, and the like.

For a message to be "silently ignored" means that is has no protocol effect but it might still be logged or reported.

## 2. General TKEY Considerations

TKEY is a meta-RR that is not stored or cached and does not appear in zone master files. It supports a variety of modes for the establishment and deletion of shared secret keying information between DNS resolvers and servers. The shared secret keying material agreed to by using TKEY is an opaque octet sequence.

The establishment of such a shared secret requires that state be maintained at both ends and the allocation of the resources to maintain such state generally requires prior mutual agreement. In the absence of willingness to provide such state, servers MUST return error NotImp or Refused for an attempt to use TKEY and resolvers are free to silently ignore any TKEY RRs they receive.

TKEY RRs are sent in the additional information section of DNS queries for type TKEY and the corresponding returned TKEY appears in the answer section of responses to such queries. Some TKEY RRs may also be spontaneously included in other responses (see Section 5.3). Type TKEY queries SHOULD NOT be flagged as recursive, and servers MUST ignore the recursion desired header bit (RD) in TKEY queries they receive. Type TKEY responses SHOULD be flagged as authoritative, and resolvers MUST ignore the authoritative answer (AA) bit in TKEY responses they receive. TKEY queries contain 1 question section entry with QTYPE TKEY, QCLASS 255 (ANY), and QNAME the same as the name of the TKEY RR.

[[[Comment: Although current practice is to have the query QNAME be the same as the name field of the TKEY RR, this does not appear to be logically necessary. Some space could be saved by using root as the QNAME.]]]

TKEY messages (DNS queries and responses) MUST be authenticated (see Section 6) for all modes except GSS-API mode, which provides its own security. In the absence of required TKEY authentication, a NotAuth error MUST be returned for a query and a reply MUST be silently ignored.

Keys established via TKEY can be treated as soft state and need not be preserved over crashes or reboots. Since DNS transactions are originated by the resolver, the resolver can simply toss keys, although it may have to go through another key exchange if it later needs one. Similarly, the server can discard keys although that will result in an error on receiving a query with a TSIG using the discarded key.

## 3. The TKEY Resource Record

The TKEY resource record (RR) is a meta-RR that has the structure shown in Table 1. Its RR type code is 249.

| Field | Format | Comment |
|---|---|---|
| NAME | domain | See description below. |
| TTYPE | u_int16_t | TKEY, MUST be 249 (0x00F9). |
| CLASS | u_int16_t | MUST be 255 (ANY, 0x00FF). |
| TTL | u_int32_t | MUST be zero. |
| RDLEN | u_int16_t | Size of RDATA. |
| RDATA: | | |
| Algorithm: | domain name | |
| Inception: | u_int32_t | |
| Expiration: | u_int32_t | |
| Mode: | u_int16_t | |
| Error: | u_int16_t | |
| Key Size: | u_int16_t | |
| Key Data: | octet-stream | |
| Other Size: | u_int16_t | |
| Other Data: | octet-stream | |

Table 1: TKEY Resource Record

Further information on these fields is provided in the subsections below.

There MUST NOT be more than one TKEY RR in a DNS message. If more than one appears in a request, a server implementing TKEY MUST return FormErr. If more than one appears in a response, they MUST all be ignored.

[[[Comment: Actually, it could be meaningful to have multiple TKEY RRs in a DNS message as long as no pair conflicted. For example, a reply could contain a key agreement TKEY RR in the answer section along with one or more key deletion TKEY RRs for other keys in the additional information section. But it seems simpler to prohibit such messages.]]]

### 3.1. Name Field

The Name field is a DNS domain name in wire encoding format. It relates to naming keys and does not normally appear as a node in the DNS name hierarchy. It MUST NOT be compressed. Its meaning differs somewhat with mode and context as explained in Section 4. In some cases where the Name field has a key name, it would be the same name as used in a TSIG RR [RFC8945] using that key.

[[[Comment: The only reason the above specifies that Name not be compressed is that there was a DNS implementation where such compression would activate a bug. This was not required in [RFC2930] and can probably be changed to allow compression of the TKEY RR Name since RR QNAMEs are normally compressible. It is domain names appearing within the TKEY RDATA, such as the Algorithm field, that are not generally compressible.]]]

Although the DNS protocol is binary clean so that any octet value should be usable in a label that is part of a domain name, to avoid possible implementation bugs and to ease user interface and debugging issues, it is RECOMMENDED that Name be composed of labels consisting of letters, digits, underscore, and hyphen and that these labels begin with a letter or digit. To indicate that no Name is present, the Name field is set to the wire encoding of the domain name of the root node, that is, the octet string consisting of a single zero value octet.

For more information on Key Names, see Section 4.1.

### 3.2. The TTL Field

The TTL field is not used in TKEY RRs. It MUST be zero to minimize the chance that a DNS implementation might not recognizing a TKEY RR as a meta-RR and would erroneously cache it. Receipt of a TKEY RR with a non-zero TTL field in a query results in a FormErr error response.

### 3.3. The RDLEN Field

The RDLEN field MUST equal the length of the RDATA section through the end of Other Data or the RR is to be considered malformed and FormErr returned if a request or the message ignored if a reply.

### 3.4. The Algorithm Field

The Algorithm name is a domain name with the same values and meaning as the Algorithm Name field in TSIG RRs (see [RFC8945]). It MUST NOT be compressed. The algorithm determines how keying material agreed to by using the TKEY RR is actually used to derive the algorithm specific secret key or keys. For example, it might need to be truncated or extended or split into multiple keys or otherwise processed.

### 3.5. The Inception and Expiration Fields

The Expiration and Inception field values specify a date and time in the form of a 32-bit unsigned number of seconds elapsed since 1 January 1970 00:00:00 UTC ignoring leap seconds, in network byte order, modulo 2**32 using ring arithmetic [RFC1982], similar to the fields with these names in the RRSIG RR [RFC4034]. In TKEY DNS messages where these fields are meaningful, their meaning depends on the mode.

Where the Inception and Expiration fields indicate a time interval, if the expiration time is before the inception time in a request, the BADTTIME error is returned in the Error field of the response.

To avoid different interpretations of the inception and expiration times in TKEY RRs, resolvers and servers exchanging them MUST have the same idea of what time it is. One way of doing this is with the Network Time Protocol [RFC5905] but that or other time synchronization used for this purpose MUST be done securely.

### 3.6. The Mode Field

The mode field specifies the general purpose of the TKEY RR, such as a key agreement method. See Section 5.1 for more information on the modes specified in this document.

### 3.7. The Error Field

The error code field is an extended RCODE [RFC6895]. In queries, it MUST be sent as zero and ignored on receipt. The following values, which may appear in replies, are used and/or specified in this document or [RFC8945]:

| Value | Description | Reference |
|---|---|---|
| 0 | - no error | |
| 1 | FormErr | [RFC1035] |
| 4 | NotImp | [RFC1035] |
| 5 | Refused | [RFC1035] |
| 9 | NotAuth | [RFC1035] |
| 16 | BADSIG | [RFC8945] |
| 17 | BADKEY | [RFC8945] |
| 18 | BADTIME | [RFC8945] |
| 19 | BADMODE | [this document] |
| 20 | BADNAME | [this document] |

Table 2: Error Codes

When the TKEY RR Error Field is non-zero in a response to a TKEY query, the DNS header RCODE field normally indicates no error. However, it is possible, if a TKEY RR is spontaneously included in a response (see Section 5.3), the TKEY RR and DNS header error fields could have unrelated non-zero error codes.

### 3.8. The Key Size and Data Fields

The Key Size Field is an unsigned 16-bit integer in network octet order. It specifies the size of the Key Data Field in octets. The meaning of the contents of the Key Data field depends on the mode and context.

### 3.9. The Other Size and Data Fields

The Other Size field is an unsigned 16-bit integer in network order which specifies the size of the Other Data field in octets. The Other Data field is intended for future expansion of the TKEY RR.

## 4. Key Naming and Key Table

DNS resolvers and servers that implement TSIG maintain a table of (1) keying material shared with other DNS servers and resolvers with which they have either been configured or successfully exchanged TKEY RRs to establish such keys and (2) information about on going attempts to establish such shared keys using TKEY. Each local table entry logically contains the information listed below. The actual structure and management of this table is a local matter as long as the behavior is consistent this document. For example, it might be accessed through one or more hash tables for rapid retrieval or have additional information such as when a key was established or be broken up into more than one table that are linked together. Some fields in this table are further described in the subsections below the table.

| Field | Format | Explanation |
|---|---|---|
| Key Name | domain name | Key name. |
| Remote Address | IPv4/IPv6 address | Address of remote DNS server/ resolver. |
| Inception | u_int32_t | Keying information inception time. |
| Expiration | u_int32_t | Keying information expiration time. |
| Key | octet-string | The opaque shared secret keying information. |
| State | unspecified | See below. |
| Algorithm | domain name | Key use algorithm. |
| Last Used | u_int32_t | The time at which a TSIG using this key was most recently sent or received and validated. |

Table 3: Key Table Entry

### 4.1. Key Name

At any DNS server or resolver only one octet string of keying material and TSIG algorithm may be in place for any particular key name. An attempt to establish agreement for a different set of keying material for an existing name returns a BADNAME error. Since a DNS TKEY reply message could be lost, it is a normal case and not an error for a server to receive a TKEY request message to establish a key that is the same as a key that the server state indicates is already established.

To avoid confusion while managing or debugging a network, it is RECOMMENDED that key names be globally unique. A key name generation strategy for resolvers and servers is described below.

#### 4.1.1. A Resolver Key Name Generation Strategy

For a TKEY RR with a non-root Name appearing in a query, the TKEY Name field SHOULD be a domain name locally unique at the resolver, less than 128 octets long in wire encoding, and meaningful to the resolver to assist in distinguishing keys and/or key agreement sessions. This length limit is suggested so that, when a resolver provided name portion is concatenated with a server provided name portion, the result will fit within the DNS protocol wire encoding name length limit of 255 octets.

#### 4.1.2. A Server Key Name Generation Strategy

For a TKEY RR appearing in a response to a query, the TKEY RR Name field SHOULD be a globally unique server assigned name unless the response indicates a TKEY error in which case the name from the query SHOULD be copied in the response.

A reasonable server key name generation strategy is as follows:

* If the key is generated by a server as the result of a query with root as its owner name, then the server SHOULD create a globally unique domain name to be the key name. A suggested method is prefixing a domain name of the server with a pseudo-random [RFC4086] label having at least 128 bits of entropy using the "file name safe" Base 64 encoding (Section 5 of [RFC4648]). For example, a8s9ZW_n3mDg-okX072pp3.serv1.example.com. If generation of a new pseudo-random name in each case is an excessive computation load or entropy drain, a serial number prefix can be added to a fixed pseudo-random name generated at DNS server start time, such as 1234.a8s9ZW_n3mDgokX072pp3_.serv1.example.com, with a new pseudo-random portion being generated periodically and on reboot.

* If the key is generated as the result of a query with a non-root name, say 789.resolv.example, then use the concatenation of that name, after deletion of its terminal root label, with a name of the server. For example, 789.resolv.example.serv1.example.com.

In the unlikely event that the unique TKEY NAME produced by whatever strategy is in use exceeds the wire encoding size limit of 255 octets, it may be shortened to fit within that limit with only an insignificant probability of losing uniqueness by replacing an initial portion of the excessively long name with the shorter encoding of a strong hash of that initial portion. For example, cut the excessively long name between labels so that the right part is no longer than 206 octets in wire encoding. Then take the prefix label or labels constituting the left part, apply SHA-256 [RFC6234] to them treated as a name in wire encoding, truncate the resulting hash to 30 octets, base32 [RFC4648] encode the result of that truncation yielding 48 octets, and add the output of the base32 encoding as a new single prefix label.

### 4.2. Remote Address

This is the address of the remote DNS server or resolver with which the key is shared or is to be shared. It MUST be possible to have table entries for more than one key for such a remote server or resolver so that, for example, a new key can be established before a key in use expires or is disabled.

### 4.3. State

This field is intended to encode an indication of the mode used in setting the key, whether the local DNS processor was filling the role of a resolver or server in the TKEY message exchange, and the status of any in progress key negotiation. The details of this field’s value are implementation dependent.

### 4.4. Last Used

This is the time signed from the most recent DNS message TSIG received and validated or sent using this Key using the same encoding as the inception and expiration TKEY fields. This field might be used to discard keys based on the least recently used or the like.

## 5. TKEY Modes

The following subsections describe the TKEY modes that are specified in this document and the messages used for each mode.

Servers and resolvers supporting this specification MUST implement the ECDH (6) mode (see Section 5.1.1), Key Deletion (5) mode (see Section 5.2.1), and TKEY Ping (8) (see Section 5.2.2) mode. Diffie-Hellman (2) mode is NOT RECOMMENDED. Resolver Assignment (4) and Server Assignment (1) modes SHOULD NOT be implemented and by definition Documentation (7) mode cannot be implemented.

A server supporting TKEY that receives a TKEY request with a mode it does not support MUST return the BADMODE error.

## 5.1. Key Agreement Modes

A resolver and a server can agree on shared secret keying material for use in TSIG through DNS requests from the resolver which are syntactically DNS queries for type TKEY and answering responses from the server. Such queries MUST have a TKEY RR in the additional information section and the response MUST have a TKEY RR in the answer section to indicate the mode in use and containing other information where required by that mode. The query and response MUST be authenticated (see Section 6) except when GSS-API mode (Section 5.1.4) is used.

The inception and expiration time in TKEY RRs with a mode intended to result in key agreement refer to a secret key validity interval, except in the case of GSS-API mode (see Section 5.1.4). The inception and expiration times in the query TKEY RR are those requested for the keying material. The inception and expiration times in the key agreement response TKEY RRs are the period the server intends to consider the keying material valid which may be a sub-interval or overlapping interval of the query specified time interval. Servers may expire keys early, so this is not a guarantee.

If the expiration time in the resolver query is in the past or if it is before the inception time, a BADTIME error MUST be returned. If the inception time in the resolver query is in the future, indicating an attempt to agree on a future key, a BADTIME error MAY be returned by the server.

> | Both DNS transaction security and DNSSEC data security originally used the SIG (type = 24) and KEY (type = 25) RRs [RFC2535]. DNSSEC was changed to use the RRSIG (type = 46) and DNSKEY (type = 48) RRs [RFC4034]; however, transaction security, including TKEY and SIG(0) [rfc2931bis], continue to use the SIG and KEY RRs.

#### 5.1.1. ECDH Exchanged Keying (Mode 6)

Elliptic Curve Diffie-Hellman (ECDH) key exchange is a means whereby two parties can derive shared secret information without requiring secrecy of the messages they exchange [Schneier] [RFC8418].

To use this mode, there need to be compatible elliptic curve public keys for the resolver and server involved [RFC6605]. However, there could be multiple keys available for each of them. The purpose of the TKEY message exchange is to select the elliptic key to be used for the resolver and the elliptic key to be used for the server, to provide nonce information, and to establish the key name and associated key value and TSIG algorithm for the resulting shared key.

A resolver sends a query for type TKEY accompanied by a TKEY RR in the additional information section specifying the ECDH exchange mode and accompanied by a KEY RR also in the additional information section specifying a resolver elliptic curve key. The TKEY RR algorithm field is set to the authentication algorithm the resolver plans to use in any TSIG RRs using the resulting key. Any "key data" provided in the TKEY in the query is used as a random [RFC4086] nonce to avoid always deriving the same keying material for the same pair of KEY RRs.

The server response contains a TKEY in its answer section with the ECDH assignment mode and echoes back the algorithm field from the query. If there is "key data" provided in this server TKEY, it is used as an additional nonce to avoid always deriving the same keying material for the same pair of KEY RRs but an anycast group of servers is only supported if such data is the same for all servers in that group. If the TKEY error field is non-zero, the query failed for the reason given. FORMERR is given if the query included no elliptic curve KEY. BADKEY is given if the query included an incompatible elliptic curve KEY.

If the response TKEY error field is zero, a server elliptic KEY RR MUST be present in the answer section of the response. The resolver supplied elliptic curve KEY RR SHOULD be echoed in the additional information section. Both parties then calculate the same shared secret quantity from the pair of elliptic curve KEY RRs used [Schneier] (provided they are compatible) and the data in the TKEY RRs, using HKDF [RFC5869] as follows where "|" indicates concatenation and the string in double quotes indicates the octet string of the ASCII [RFC0020] for that string without any terminating zero octet or leading length octet or surrounding quotes:

```
IKM = shared secret from ECDH with resolver and server keys
salt = resolver-TKEY-data | server-TKEY-data
info = "IETF-TKEY-ECDH"

OKM = HKDF-Expand(HMAC-Hash(salt, IKM), info, L)
```

SHA-256 [RFC6234] is used in the HMAC-Hash [RFC2104]. L is the length of the keying material needed for use in the Algorithm specified. and OKM is the output keying material to use with TSIG.

#### 5.1.2. Server Assigned Keying (Mode 1, Deprecated)

A mode was specified in the previous [RFC2930] to this document whereby a server can assign keying to be shared with the resolver. So far as is known, this mode was never implemented and SHOULD NOT be implemented now. It is not specified in this document.

#### 5.1.3. Resolver Assigned Keying (Mode 4, Deprecated)

A mode was specified in the previous [RFC2930] to this document whereby a server can accept a resolver assigned key. So far as is known, this mode was never implemented and SHOULD NOT be implemented now. It is not specified in this document.

#### 5.1.4. GSS-API Establishment (Mode 3)

This mode is described in "Generic Security Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)" [RFC3645] which should be consulted for the full description. Basically, the resolver and server can exchange queries and responses for type TKEY with a TKEY RR specifying the GSS-API mode in the additional information section and a GSS-API token in the key data portion of the TKEY RR.

Any issues of possible encryption of parts the GSS-API token data being transmitted are handled by the GSS-API level. In addition, the GSS-API level provides its own authentication so that this mode of TKEY query and response MAY be, but does not need to be, authenticated with a TSIG RR [RFC8945] or SIG(0) RR [rfc2931bis].

The inception and expiration times in a GSS-API mode TKEY RR are ignored.

#### 5.1.5. Diffie-Hellman Exchanged Keying (Mode 2, Deprecated)

The use of Diffie-Hellman Exchanged Keying mode is NOT RECOMMENDED for the reasons given in Appendix A which also contains the specification for that mode for compatibility with old TKEY implementations. Use of ECDH Exchanged Keying is RECOMMENDED as an alternative (see Section 5.1.1).

### 5.2. Other Modes

The TKEY modes specified in the subsections below do not provide for agreement on a key but provide other functions.

#### 5.2.1. TKEY Deletion (Mode 5)

To avoid attempted reliance in requests on keys no longer in effect, servers MUST implement TKEY Deletion mode whereby the server "discards" a key on receipt from a resolver of an authenticated TKEY Deletion mode TKEY RR with that key's name. To be effective, the TKEY MUST also have an inception time no later than the inception of the key to be deleted and an expiration time no earlier than the expiration time of the key to be deleted. This time restriction is intended to minimize potential insecurities due to replaying deletion mode TKEY RRs.

If the server has no record of a key with that name, it returns BADNAME in the TKEY Error field in the reply. This error condition may be due to the server having discarded the key. If it has a key with that name but with a validity period extending outside that in the deletion request, it returns a BADTIME error.

Key deletion TKEY queries MUST be authenticated (see Section 6). This authentication MAY be a TSIG RR using the key to be deleted.

For ECDH or Diffie-Hellman keys, the server SHOULD discard all active state associated with the key name in its key table.

Keys may also be deleted through spontaneous inclusion of a deletion mode TKEY RR in a response as specified in Section 5.3.

#### 5.2.2. TKEY Ping (Mode 8)

The TKEY Ping Mode is intended for use as a test of basic TKEY plumbing and to check the synchronization of the resolver and server clocks. It also provides a means for a resolver to determine if TKEY is implemented at a server without changing the key storage state.

In a TKEY Ping mode query, the Name and Algorithm fields SHOULD be set to root to save space and are ignored by the server. The Inception field MUST be set to the time the query is constructed according to the resolver's clock. The Expiration field MUST be set to zero in a query and is ignored on receipt. A resolver MAY include a sequence number or identification information or whatever else it might find useful in the Key field. And the Other Data field SHOULD be sent empty and ignored by the server.

A server implementing TKEY responds with a Ping Mode TKEY RR in the answer section of its response copied from the query except that it MUST set the Expiration field to the value of the server's clock when the reply is constructed even if the response also indicated an error such as BADTIME.

#### 5.2.3. Documentation Mode (Mode 7)

This mode is assigned as a TKEY Mode to use in examples or documentation. A server responds with BADMODE as the TKEY error if it receives a TKEY RR indicating this mode. Mode 7 will never be assigned any other use.

### 5.3. Spontaneous Server Inclusion

A DNS server MAY include a deletion mode (mode 5) TKEY RR spontaneously in the additional information portion of any DNS query response. To avoid confusion, this SHOULD NOT be done unless the server has reason to believe that the resolver supports TKEY and SHOULD only be done if the server has deleted the corresponding secret key. This technique may be specified or allowed for modes other than deletion in the future. A disadvantage of this technique is that there is no way for the server to get any error or success indication back and, in the case of UDP transport, no way to even know if the DNS response reached the resolver.

Such a response MUST be authenticated for the TKEY to be effective. If it is not authenticated, the resolver should ignore any included TKEY RR. Failure by a client to receive or properly process such additional information in a response would mean that the client might use a key that the server had discarded in a TSIG and would then get an error indication.

## 6. TKEY Message Authentication

Unless otherwise specified, all TKEY queries and responses MUST be authenticated. A server receiving a TKEY query that is not authenticated but needs to be authenticated returns a NotAuth error in the returned TKEY Error field. A resolver receiving a TKEY reply that needs to be authenticated but is not authenticated silently ignores the TKEY RR.

To avoid replay attacks, it is necessary that a TKEY response or query not be valid if replayed on the order of 2**32 second (about 136 years), or a multiple thereof, later. To accomplish this, the keying material used in any TSIG or SIG(0) RR that authenticates a TKEY message MUST NOT have a lifetime of more than 2**31 - 1 seconds (about 68 years). Thus, on attempted replay, the authenticating TSIG or SIG(0) RR would not be verifiable due to key expiration and a replay would fail.

There are two methods of authentication as specified below.

### 6.1. Secret Key Authentication

If a shared secret key is in place between the resolver and server, then TKEY queries and response can be authenticated with TSIG [RFC8945]. So, for example, an existing shared secret key could be used to authenticate a TKEY exchange that sets up a new key before rolling over to that new key.

### 6.2. Public Key Authentication

As an alternative to secret key authentication, public key authentication can be used with the SIG(0) RR as specified in [rfc2931bis].

## 7. IANA Considerations

This section is to be interpreted as provided in [RFC8126].

### 7.1. Reference Update

IANA is requested to update all reference to [RFC2930] in the DNS Parameters Registry Group to reference [this document].

### 7.2. Existing Assignments

The following assignments have already been made:

* RR Type 249 for TKEY.

* Extended RCODE Error values of 19, 20, and 21 as listed in Section 3.7.

### 7.3. TSIG/TKEY Registry Group

IANA is requested to rename the existing "Secret Key Transaction Authentication for DNS (TSIG) Algorithm Names" registry to be the "DNS TSIG and TKEY Parameters" registry group and include within it the existing "TSIG Algorithm Names" registry and the new registry created below.

#### 7.3.1. TKEY Modes Registry

IANA is requested to create a "TKEY Modes" Registry as follows:

Name: TKEY Modes
Reference: [this document]
Registration Procedures:
0x0000 - reserved
0x0001-0x0FFF - standards action
0x1000-0xFFEF - specification required
0xFFF0-0xFFFE - experimental use
0xFFFF - reserved

Initial contents as follows:

| Value | Description | Implementation | Reference |
|---|---|---|---|
| 0x0000 | - reserved | - | |
| 0x0001 | Server Assignment | SHOULD NOT | Section 5.1.2, [this document] |
| 0x0002 | Diffie-Hellman exchange | SHOULD NOT | Appendix A, [this document] |
| 0x0003 | GSS-API Negotiation | MAY | Section 5.1.4, [this document] |
| 0x0004 | Resolver Assignment | SHOULD NOT | Section 5.1.3, [this document] |
| 0x0005 | Key Deletion | MUST | Section |

| | | | |
|---|---|---|---|
| | | | 5.2.1, [this document] |
| 0x0006 | ECDH Agreement | MUST | Section 5.1.1 [this document] |
| 0x0007 | Documentation/ example | N/A | Section 5.2.3, [this document] |
| 0x0008 | TKEY Ping | MUST | Section 5.2.2, [this document] |
| 0x0009-0x0FFF | - unassigned | - | |
| 0x1000-0xFFEF | - unassigned | - | |
| 0xFFF0-0xFFFE | - experimental use | - | |
| 0xFFFF | - reserved | - | |

Table 4

## 8. Security Considerations

This specification is concerned with the secure establishment of shared secret keys between DNS clients and servers in support of TSIG [RFC8945].

Secret keys agreed to using TKEY and the private keys corresponding to the public key belonging to DNS resolvers and servers are very sensitive information. All practical steps should be taken to protect them on every host on which they are stored. Generally, such hosts need to be physically protected. If they are shared machines, great care should be taken so that unprivileged processes have no access to keying material. Resolvers sometimes run unprivileged, which means all users of a host might be able to see whatever configuration data are used by the resolver.

Protection against denial of service via the use of TKEY is not provided. DNS servers and resolve should implement appropriate rate limiting.

More TBD

## 9. Normative References

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[rfc2931bis] Eastlake, D. and J. Stenstam, "Domain Name System (DNS) Public Key Based Request and Transaction Authentication ( SIG(0) )", work in process, <https://datatracker.ietf.org/doc/draft-eastlake-dnsop-rfc2931bis-tsigzero/>.

[RFC3645] Kwan, S., Garg, P., Gilroy, J., Esibov, L., Westhead, J., and R. Hall, "Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG)", RFC 3645, DOI 10.17487/RFC3645, October 2003, <https://www.rfc-editor.org/info/rfc3645>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6605] Hoffman, P. and W.C.A. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, DOI 10.17487/RFC6605, April 2012, <https://www.rfc-editor.org/info/rfc6605>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8418] Housley, R., "Use of the Elliptic Curve Diffie-Hellman Key Agreement Algorithm with X25519 and X448 in the Cryptographic Message Syntax (CMS)", RFC 8418, DOI 10.17487/RFC8418, August 2018, <https://www.rfc-editor.org/info/rfc8418>.

## 10. Informative References

[Schneier] Schneier, B., "Applied Cryptography Second Edition: protocols, algorithms, and source code in C", ISBN 0-471-11709-9, 1966.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <https://www.rfc-editor.org/info/rfc1995>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2535] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, DOI 10.17487/RFC2535, March 1999, <https://www.rfc-editor.org/info/rfc2535>.

[RFC2539] Eastlake 3rd, D., "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)", RFC 2539, DOI 10.17487/RFC2539, March 1999, <https://www.rfc-editor.org/info/rfc2539>.

[RFC2930] Eastlake 3rd, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, DOI 10.17487/RFC2930, September 2000, <https://www.rfc-editor.org/info/rfc2930>.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <https://www.rfc-editor.org/info/rfc3007>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <https://www.rfc-editor.org/info/rfc6895>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <https://www.rfc-editor.org/info/rfc8945>.

[RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <https://www.rfc-editor.org/info/rfc9499>.

## Appendix A. Diffie-Hellman Exchanged Keying Details

TKEY Mode 2, Diffie-Hellman Exchanged Keying, is deprecated for the reasons given below. It SHOULD NOT be used unless needed for compatibility with old TKEY implementations.

1. The mixing function used does not meet current cryptographic standards because it uses MD5 [RFC6151].

2. The ad hoc method of combining the DH derived shared value with various nonces is inferior to the HKDF [RFC5869] method used with the ECDH TKEY mode specified in this document.

Diffie-Hellman (DH) key exchange is means whereby two parties can derive some shared secret information without requiring any secrecy of the messages they exchange [Schneier]. When this mode was originally specified, it was assumed that RSA public/private keys would be used. Provisions have been made for the storage of "DH" (RSA) public keys in the DNS [RFC2539].

A resolver sends a query for type TKEY accompanied by a TKEY RR in the additional information section specifying the Diffie-Hellman mode and accompanied by a KEY RR also in the additional information section specifying a resolver public key. The TKEY RR algorithm field is set to the authentication algorithm the resolver plans to use. The "key data" provided in the TKEY is used as a random [RFC4086] nonce to avoid always deriving the same keying material for the same pair of DH KEYs.

The server response contains a TKEY in its answer section with the Diffie-Hellman mode. The "key data" provided in this TKEY is used as an additional nonce to avoid always deriving the same keying material for the same pair of DH KEYs. If the TKEY error field is non-zero, the query failed for the reason given. FORMERR is given if the query included no DH KEY and BADKEY is given if the query included an incompatible DH KEY.

If the response TKEY error field is zero, the resolver supplied KEY RR SHOULD be echoed in the additional information section and a server Diffie-Hellman KEY RR MUST be present in the answer section of the response. Both parties can then calculate the same shared secret quantity from the pair of public keys used [Schneier] (provided these keys are compatible) and the data in the TKEY RRs. The TKEY RR data is mixed with the DH result as follows:

```
keying material =
     XOR ( DH value, MD5 ( query data | DH value ) |
                     MD5 ( server data | DH value ) )
```

Where XOR is the bitwise exclusive-OR operation and "|" is octet-string concatenation. The shorter of the two operands to XOR is octet-wise left justified and padded with zero-valued octets to match the length of the other operand. "DH value" is the Diffie-Hellman value derived from the KEY RRs. Query data and server data are the values sent in the TKEY RR data fields. These "query data" and "server data" nonces are suffixed by the DH value, digested by MD5, the results concatenated, and then XORed with the DH value.

The inception and expiration times in the query TKEY RR are those requested for the keying material. The inception and expiration times in the response TKEY RR are the maximum period the server will consider the keying material valid. Servers may pre-expire keys, so this is not a guarantee.

## Appendix B. Changes from [RFC2930]

This document incorporates the following significant changes from [RFC2930]:

* Addition of the ECDH Key Agreement mode (#6), the Documentation mode (#7), and the TKEY Ping mode (#8).

* Deprecation of Diffie-Hellman Exchanged Keying (mode #2).

* Deprecation of and dropping of the specifications for Server Assignment mode (#1) and Resolver Assignment (#4) node.

* Additional material including that concerned with authentication of TKEY RRs.

* Editorial changes including some re-organization.

## Acknowledgements

The comments and suggestions of the following are gratefully acknowledged:

tbd

The comments and suggestions of the following persons were incorporated into RFC 2930, which was the previous version of this document, and are gratefully acknowledged:

Olafur Gudmundsson, Stuart Kwan, Ed Lewis, Erik Nordmark, and Brian Wellington.

Authors' Addresses

Donald E. Eastlake 3rd
Independent
2386 Panoramic Circle
Apopka, Florida 32703
United States of America
Phone: +1-508-333-2270
Email: d3e3e3@gmail.com

Mark Andrews
Internet Systems Consortium
Email: marka@isc.org