INTERNET-DRAFT
Obsoletes: 2930
Intended Status: Proposed Standard
Expires: April 23, 2023

D. Eastlake
Futurewei Technologies

October 24, 2022

Secret Key Agreement for DNS (TKEY Resource Record)

<draft-eastlake-dnsop-rfc2930bis-tkey-00.txt>

Abstract

RFC 8945 provides a means of efficiently authenticating Domain Name System (DNS) protocol messages using shared secret keys via the TSIG resource record (RR). However, it provides no mechanism for setting up such keys other than manual configuration. This document describes a TKEY RR that can be used in a number of different modes to establish shared secret keys between a DNS resolver and server. This document obsoletes RFC 2930.

Status of This Document

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

Distribution of this document is unlimited. Comments should be sent to the author.

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

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

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html. The list of Internet-Draft Shadow Directories can be accessed at

https://www.ietf.org/shadow.html.

Table of Contents

1. Introduction

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

[RFC8945] provides a means of efficiently authenticating DNS messages using shared secret keys via the TSIG resource record (RR) but provides no mechanism for setting up such keys other than manual configuration. This document specifies a TKEY RR that can be used in a number of different modes to establish and delete such shared secret keys between a DNS resolver and server.

Note that TKEY established keying material and TSIGs that use it are associated with DNS servers or resolvers. They are not associated with zones. They may be used to authenticate queries and responses but they do not provide zone based DNS data origin or denial 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.

In all cases herein, the term "resolver" includes that part of a server which may make full and incremental [RFC1995] zone transfer queries, forwards recursive queries, and the like.

1.2 Overview of Contents

Section 2 below specifies the TKEY RR and provides a description of and considerations for its constituent fields.

Section 3 describes general principles of operations with TKEY.

Section 4 discusses key agreement and deletion via DNS requests with the Query opcode for RR type TKEY. This method is applicable to all currently defined TKEY modes, although in some cases it is not what would intuitively be called a "query".

Section 5 discusses spontaneous inclusion of TKEY RRs in responses by servers. This is currently used only for key deletion.

Section 6 describes encryption methods for transmitting secret key information. In this document these are used only for the server assigned mode and the resolver assigned mode.

Section 7 covers IANA Considerations in assignment of TKEY modes.

Finally, Section 8 provides a Security Considerations section.

2. The TKEY Resource Record

The TKEY resource record (RR) has the structure given below. Its RR type code is 249.

```
Field       Type         Comment
-----       ----         -------
NAME         domain      see description below
TTYPE        u_int16_t   TKEY = 249
CLASS        u_int16_t   ignored, SHOULD be 255 (ANY)
TTL          u_int32_t   ignored, SHOULD 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
```

2.1 The Name Field

The Name field is a DNS domain name in wire encoding format. It relates to naming keys. Its meaning differs somewhat with mode and context as explained in subsequent sections. Although the DNS protocol in 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 ease user interface and debugging issues, it is RECOMMENDED that Name be composed of labels consisting of letters, digits, underscore, and hyphen. 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 byte string consisting of a single zero value byte.

At any DNS server or resolver only one octet string of keying material may be in place for any particular key name. An attempt to establish another set of keying material for an existing name returns a BADNAME error.

For a TKEY RR appearing in a query with a non-root Name, 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 is concatenated with a server provided name portion,

the result will fit within the DNS protocol wire encoding name length limit of 255 octets. For TKEY RR(s) appearing in a response to a query, the TKEY RR Name field SHOULD be a globally unique server assigned name.

A reasonable server key naming 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, by 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_n3mDgokX072pp3_.server1.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 2001.a8s9ZW_n3mDgokX072pp3_.server1.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.resolver.example.net, then use the concatenation of that name, after deletion of its terminal root label, with a name of the server. For example, 789.resolver.example.net.server1.example.com.
>
> If 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 on the left, 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.

2.2 The TTL Field

The TTL field is meaningless in TKEY RRs. It SHOULD always be zero to minimize the chance that a DNS implementation not recognizing the TKEY RR would cache such an RR.

2.3 The Algorithm Field

The algorithm name is in the form of a domain name with the same meaning as in [RFC8945]. The algorithm determines how the secret keying material agreed to by using the TKEY RR is actually used to derive the algorithm specific key or keys. For example, it might need to be truncated or extended or split into multiple keys.

2.4 The Inception and Expiration Fields

The inception and expiration times are in number of seconds since the beginning of 1 January 1970 GMT ignoring leap seconds treated as modulo 2**32 using ring arithmetic [RFC1982]. In messages between a DNS resolver and a DNS server where these fields are meaningful, they are either the requested validity interval for the keying material asked for or specify the validity interval of keying material provided.

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 NTP protocol [RFC5905] but that or any other time synchronization used for this purpose MUST be done securely.

2.5 The Mode Field

The mode field specifies the general scheme for key agreement or the purpose of the TKEY DNS message. Servers and resolvers supporting this specification MUST implement the ECDH key agreement mode (#6), key deletion mode (#5), and TKEY ping (#8). All other modes are OPTIONAL. A server supporting TKEY that receives a TKEY request with a mode it does not support returns the BADMODE error. See Section 7 for initial mode field value assignments.

2.6 The Error Field

The error code field is an extended RCODE. The following values are defined:

Value	Description
0	- no error
1-15	a non-extended RCODE
16	BADSIG (tsig)
17	BADKEY (tsig)
18	BADTIME (tsig)
19	BADMODE
20	BADNAME
21	BADALG

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

2.7 The Key Size and Data Fields

The key data size field is an unsigned 16-bit integer in network order which specifies the size of the key exchange data field in octets. The meaning of this data depends on the mode.

2.8 The Other Size and Data Fields

The Other Size and Other Data fields are not used in this specification but may be used in future extensions. 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 rejected.

3. General TKEY Considerations

TKEY is a meta-RR that is not stored or cached in the DNS and does not appear in zone files. It supports a variety of modes for the establishment and deletion of shared secret keys information between DNS resolvers and servers. The establishment of such a shared key requires that state be maintained at both ends and the allocation of the resources to maintain such state may require mutual agreement. In the absence of willingness to provide such state, servers MUST return errors such as NOTIMP or REFUSED for an attempt to use TKEY and resolvers are free to ignore any TKEY RRs they receive.

The shared secret keying material developed by using TKEY is an opaque octet sequence. The means by which this shared secret keying material, exchanged via TKEY, is actually used in any particular TSIG algorithm is algorithm dependent and is defined in connection with that algorithm. For example, see [RFC2104] for how TKEY agreed shared secret keying material can be used with HMAC and a strong hash.

There MUST NOT be more than one TKEY RR in a DNS query or response. If more than one appears in a query, NOTIMP or FORMERR is returned.

Except for GSS-API mode, TKEY responses MUST always have DNS transaction authentication to protect the integrity of any keying data, error codes, etc. This authentication MUST use a previously established secret (TSIG) or public (SIG(0)) key and MUST NOT use any key that the response to be verified is itself providing.

TKEY queries MUST be authenticated for all modes except GSS-API and, under some circumstances, server assignment mode. If the query for a server assigned key is for a key to assert some privilege, such as update authority, then the query must be authenticated to avoid spoofing. However, if the key is just to be used for transaction security, then spoofing will lead at worst to denial of service. Query authentication SHOULD use an established secret (TSIG) key authenticator if available. Otherwise, it MUST use a public (SIG(0)) key signature. It MUST NOT use any key that the query is itself providing.

In the absence of required TKEY authentication, a NOTAUTH error MUST be returned.

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 will not be verifiable due to key expiration and the

replay will fail.

4. Resolver Query TKEY Modes

One method for a resolver and a server to agree about shared secret keying material for use in TSIG is through DNS requests from the resolver which are syntactically DNS queries for type TKEY. Such queries MUST be accompanied by a TKEY RR in the additional information section to indicate the mode in use and accompanied by other information where required.

Type TKEY queries SHOULD NOT be flagged as recursive, and servers MUST ignore the recursive header bit in TKEY queries they receive.

4.1 Query for Server Assigned Keying

Optionally, the server can assign keying to the resolver in response to a Server Assignment Mode (#1) query. It is sent to the resolver encrypted under a resolver public key. See section 6 for description of encryption methods.

A resolver sends a query for type TKEY accompanied by a TKEY RR specifying the "server assignment" mode and a resolver KEY RR to be used in encrypting the response, both in the additional information section. The TKEY algorithm field is ignored and SHOULD be set to root to minimize message size. It is RECOMMENDED that any "key data" provided in the query TKEY RR by the resolver be strongly mixed by the server with server generated randomness [RFC4086] to derive the keying material to be sent which should be at least 32 octets long. The KEY RR that appears in the query need not be accompanied by a SIG(KEY) RR. If the query is authenticated by the resolver with a TSIG RR [RFC8945] or SIG(0) RR and that authentication is verified, then any SIG(KEY) provided in the query SHOULD be ignored. The KEY RR in such a query SHOULD have a name that corresponds to the resolver but it is only essential that it be a public key for which the resolver has the corresponding private key so it can decrypt the response data.

The server response contains a TKEY RR in its answer section with the Server Assignment Mode and echoes the KEY RR provided in the query in its additional information section.

If the response TKEY error field is zero, the key data portion of the response answer TKEY RR will be the server assigned keying data encrypted under the public key in the resolver provided KEY RR. In this case, the owner name of the response answer TKEY RR will be the server assigned name of the key.

If the error field of the response TKEY is non-zero, the query failed

for the reason given. FORMERR is given if the query specified no

encryption key.

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

The resolver KEY RR MUST be authenticated, through the authentication of this query with a TSIG or SIG(0) or the signing of the resolver KEY with a SIG(KEY). Otherwise, an attacker can forge a resolver KEY for which they know the private key, and thereby the attacker could obtain a valid shared secret key from the server.

4.2 Diffie-Hellman Exchanged Keying (Deprecated)

The use of this mode (#2) is NOT RECOMMENDED for the following two reasons but the specification is still included in Appendix A in case an implementation is needed for compatibility with old TKEY implementations. See Section 4.6 on ECDH Exchanged Keying.

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

RSA keys must be excessively long to achieve levels of security required by current standards.

4.3 Query for GSS-API Establishment

This mode (#3) is described in [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 do not need to be, authenticated with TSIG RR or SIG(0) RR.

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

4.4 Query for Resolver Assigned Keying

Optionally, a server can accept a resolver assigned key. The keying material MUST be encrypted under a server key for protection in transmission as described in Section 6.

The resolver sends a TKEY query with a Resolver Assignment Mode (#4) TKEY RR that specifies the encrypted keying material and a KEY RR specifying the server public key used to encrypt the data, both in the additional information section. The name of the key and the keying data are completely controlled by the sending resolver so a globally unique key name SHOULD be used. The KEY RR used MUST be one for which the server has the corresponding private key, or it will not be able to decrypt the keying material and will return a FORMERR. It is also important that no untrusted party (preferably no other party than the server) has the private key corresponding to the KEY RR because, if they do, they can capture the messages to the server, learn the shared secret, and spoof valid TSIGs.

The query TKEY RR inception and expiry give the time period the querier intends to consider the keying material valid. The server can return a lesser time interval to advise that it will not maintain state for that long and can pre-expire keys in any case.

This type of query MUST be authenticated with a TSIG or SIG(0). Otherwise, an attacker can forge a resolver assigned TKEY query, and thereby the attacker could specify a shared secret key that would be accepted, used, and honored by the server.

4.5 Query for TKEY Deletion

Keys established via TKEY can be treated as soft state. 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.

To avoid attempted reliance in requests on keys no longer in effect, servers MUST implement key deletion node (#5) whereby the server "discards" a key on receipt from a resolver of an authenticated delete request for a TKEY RR with the key's name. If the server has no record of a key with that name, it returns BADNAME.

Key deletion TKEY queries MUST be authenticated. This authentication MAY be a TSIG RR using the key to be deleted.

For querier assigned keys and ECDH or Diffie-Hellman keys, the server

SHOULD "discard" all active state associated with the key. For server assigned keys, the server MAY simply mark the key as no longer retained by the client and may re-send it in response to a future query for server assigned keying material.

4.6 Query for ECDH Exchanged Keying

Elliptic Curve Diffie-Hellman (ECDH) key exchange is a means whereby two parties can derive shared secret information without requiring any secrecy of the messages they exchange [Schneier]. [RFC8418] [RFC6605]

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. 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 KEY RRs.

The server response contains a TKEY in its answer section with the ECDH assignment 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 KEY RRs. 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 and BADKEY is given if the query included an incompatible elliptic curve KEY.

If the response TKEY error field is zero, the resolver supplied elliptic curve KEY RR SHOULD be echoed in the additional information section and a server elliptic 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 elliptic curve KEY RRs used [Schneier] (provided they are compatible and the data in the TKEY RRs. The TKEY RR data is mixed with the DH result as follows:

```
keying material =
     XOR ( ECDH value, SHA-256 ( query data | ECDH value ),
                       SHA-256 ( server data | ECDH value ) )
```

Where XOR is an exclusive-OR operation and "|" is byte-stream concatenation. The shorter operands to XOR are byte-wise left justified and padded with zero-valued bytes to match the length of the other. "ECDH value" is the shared secret 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 ECDH value, digested by SHA-256, and then XORed with

the ECDH value.

The inception and expiry times in the query TKEY RR are those requested for the keying material. The inception and expiry 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.

4.7 Example Mode

The value 7 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.

4.8 TKEY Ping

The TKEY Ping Mode (#8) is intended for use as a test of basic TKEY plumbing. It also provides a means for a querier to determine if TKEY is implemented at a server without changing the key storage state. A server implementing TKEY MUST simply echo back a Ping Mode TKEY RR in its respose.

5. Spontaneous Server Inclusion

A DNS server may include a TKEY RR spontaneously as additional information in responses. This SHOULD only be done if the server knows the querier understands TKEY and has this option implemented. This technique can be used to delete a key and may be specified for other modes defined 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, no way to even know if the DNS response reached the resolver.

5.1 Spontaneous Server Key Deletion

A server can optionally tell a client that it has deleted a secret key by spontaneously including a TKEY RR in the additional information section of a response with the key's name and specifying the key deletion mode (#5). Such a response SHOULD be authenticated. If authenticated, it "deletes" the key with the given name. The inception and expiry times of the delete TKEY RR are ignored. 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 and would then get an error indication.

For server assigned and ECDH keys, the client MUST "discard" active state associated with the key. For querier assigned keys, the querier MAY simply mark the key as no longer retained by the server and may re-send it in a future query specifying querier assigned keying material.

6. Methods of Encryption

For the server assigned and resolver assigned key agreement modes, the keying material is sent within the key data field of a TKEY RR encrypted using the public key in an accompanying KEY RR [RFC4034].

If this KEY RR is for a public key algorithm where the public and private keys can be used for encryption and the corresponding decryption which recovers the originally encrypted data. The KEY RR SHOULD correspond to a name for the decrypting resolver/server such that the decrypting process has access to the corresponding private key to decrypt the data. The secret keying material being sent will generally be fairly short, usually not more than 256 bits, because that is adequate for very strong protection with modern keyed hash or symmetric algorithms.

> If the KEY RR specifies the RSA algorithm, then the keying material is encrypted as per the description of RSAES-PKCS1-v1_5 encryption in PKCS#1 [RFC8017]. The secret keying material being sent is directly RSA encrypted in PKCS#1 format. It is not "enveloped" under some other symmetric algorithm. In the unlikely event that the keying material will not fit within one RSA modulus of the chosen public key, additional RSA encryption blocks are included. The length of each block is clear from the public RSA key specified and the RSAES-PKCS1-v1_5 padding makes it clear what part of the encrypted data is actually keying material and what part is formatting or the required at least eight bytes of random [RFC4086] padding.

If the KEY RR accompanying the TKEY RR is for a key agreement algorithm where the public and private keys cannot be used for encryption/decryption, then the data to be encrypted is "enveloped" as specified below. The Key Data Field is structured as follows:

Subfield	Type	Comment
Wrapping	u_int8_t	Key Wrapping Algorithm
WKey Size	u_int8_t	Size of Wrapping Key
Key	octet-stream	Wrapped keying material

For server assignment keying, a shared secret key is derived from the public key in the KEY RR sent with the query and the KEY RR included in the corresponding response. This key is used to wrap the keying material being transmitted but may need to be truncated to the specified WKey Size. In addition, some Key Wrapping Algorithms take a variety of key sizes so the wrapping key size must be specified.

For resolver assignment keying TBD

7. IANA Considerations

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

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 2.6.

IANA is requested to create a TKEY Mode Registry on the Domain Name Sysmtem (DNS) Parameters web page 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

Value	Description	Reference
0x0000	- reserved	
0x0001	server assignment	[this document]
0x0002	Diffie-Hellman exchange	[this document]
0x0003	GSS-API negotiation	[this document]
0x0004	resolver assignment	[this document]
0x0005	key deletion	[this document]
0x0006	ECDH exchange	[this document]
0x0007	documentation/example	[this document]
0x0008	TKEY ping	[this document]
0x0009 - 0x0FFF	- standards action	
0x1000 - 0xFFEF	- specification required	
0xFFF0 - 0xFFFE	- experimental use	
0xFFFF	- reserved	

IANA is requested to create a Key Wrapping Algorithm Registry on a new web page as follows:

Name: Key Wrapping Algorithms
Reference: [this document]
Registration Procedure: IETF Consensus

Value	Description	Reference
0x00	- reserved	
0x01	Triple-DES	[RFC3217]
0x02	RC2	[RFC3217]
0x03	AES-pad	[RFC5649]
0x04-0xFE	unassigned	
0xFF	- reserved	

8. Security Considerations

The entirety of this specification is concerned with the secure establishment of a shared secret between DNS clients and servers in support of TSIG [RFC8945].

Protection against denial of service via the use of TKEY is not provided.

Appendix A: Diffie-Hellman Exchanged Keying

This TKEY Mode (#1) is deprecated for the reasons given in Section 4.1. It SHOULD NOT be used.

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]. Provisions have been made for the storage of DH 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 Diffie-Hellman 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 Diffie-Hellman 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 Diffie-Hellman (DH) keys used [Schneier] (provided these DH keys use the same generator and modulus) 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 an exclusive-OR operation and "|" is byte-stream concatenation. The shorter of the two operands to XOR is byte-wise left justified and padded with zero-valued bytes 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 expiry times in the query TKEY RR are those requested for the keying material. The inception and expiry 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.

Normative References

[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>.

[RFC3217] - Housley, R., "Triple-DES and RC2 Key Wrapping", RFC 3217, DOI 10.17487/RFC3217, December 2001, <https://www.rfc-editor.org/info/rfc3217>.

[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>.

[RFC5649] - Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <https://www.rfc-editor.org/info/rfc5649>.

[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. 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>.

Informative References

[Schneier] - Bruce Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 1996, John Wiley and Sons

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8017] - Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[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>.

Acknowledgments

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.

Author's Address

Donald E. Eastlake 3rd
Futurewei Technologies, Inc.
2386 Panoramic Circle
Apopka, FL 32703 USA

Telephone: +1 508 333 2270
email: d3e3e3@gmail.com