Network Working Group
Request for Comments: 2930
Category: Standards Track

D. Eastlake, 3rd
Motorola

September 2000

Secret Key Establishment for DNS (TKEY RR)

Status of this Memo

Copyright Notice

Abstract

[RFC 2845] provides a means of authenticating Domain Name System (DNS) queries and responses using shared secret keys via the Transaction Signature (TSIG) resource record (RR). However, it provides no mechanism for setting up such keys other than manual exchange. This document describes a Transaction Key (TKEY) RR that can be used in a number of different modes to establish shared secret keys between a DNS resolver and server.

Acknowledgments

The comments and ideas of the following persons (listed in alphabetic order) have been incorporated herein and are gratefully acknowledged:

Olafur Gudmundsson (TIS)

Stuart Kwan (Microsoft)

Ed Lewis (TIS)

Erik Nordmark (SUN)

Brian Wellington (Nominum)

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 [RFC 1034, 1035]. It has been extended to provide for public key security and dynamic update [RFC 2535, RFC 2136]. Familiarity with these RFCs is assumed.

[RFC 2845] 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 exchange. 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 [RFC 2535].

Certain modes of TKEY perform encryption which may affect their export or import status for some countries. The affected modes specified in this document are the server assigned mode and the resolver assigned mode.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

In all cases herein, the term "resolver" includes that part of a server which may make full and incremental [RFC 1995] zone transfer queries, forwards recursive queries, etc.

1.1 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 which 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 the required 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
    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  undefined by this specification
```

2.1 The Name Field

The Name field relates to naming keys. Its meaning differs somewhat with mode and context as explained in subsequent sections.

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 at a server for an existing name returns a BADNAME error.

For a TKEY with a non-root name appearing in a query, the TKEY RR name SHOULD be a domain 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. For TKEY(s) appearing in a response to a query, the TKEY RR name SHOULD be a globally unique server assigned domain.

A reasonable key naming strategy is as follows:

> If the key is generated 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 suffixing a pseudo-random [RFC 1750] label with a domain name of the server. For example 89n3mDgX072pp.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 an DNS server start time, such as 1001.89n3mDgX072pp.server1.example.com.

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 with a name of the server. For example 789.resolver.example.net.server1.example.com.

2.2 The TTL Field

The TTL field is meaningless in TKEY RRs. It SHOULD always be zero to be sure that older DNS implementations do not cache TKEY RRs.

2.3 The Algorithm Field

The algorithm name is in the form of a domain name with the same meaning as in [RFC 2845]. The algorithm determines how the secret keying material agreed to using the TKEY RR is actually used to derive the algorithm specific key.

2.4 The Inception and Expiration Fields

The inception time 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 [RFC 1982]. 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 [RFC 2030] 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 Diffie-Hellman key agreement mode and the key deletion mode for queries. 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. The following values of the Mode octet are defined, available, or reserved:

Value	Description
0	- reserved, see section 7
1	server assignment
2	Diffie-Hellman exchange
3	GSS-API negotiation
4	resolver assignment
5	key deletion
6-65534	- available, see section 7
65535	- reserved, see section 7

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 a plain 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 [RFC 2104] for how TKEY agreed shared secret keying material is used in the HMAC-MD5 algorithm or other HMAC algorithms.

There MUST NOT be more than one TKEY RR in a DNS query or response.

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) [RFC 2931]) 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. In particular, 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 then 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. Exchange via Resolver Query

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 MAY ignore the recursive header bit in TKEY queries they receive.

4.1 Query for Diffie-Hellman Exchanged Keying

Diffie-Hellman (DH) key exchange is a 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 [RFC 2539].

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 [RFC 1750] 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 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 will also 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.

4.2 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 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 and Diffie-Hellman keys, the server MUST truly "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.

4.3 Query for GSS-API Establishment

This mode is described in a separate document under preparation which should be seen 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 [RFC 2931].

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

4.4 Query for Server Assigned Keying

Optionally, the server can assign keying for the resolver. 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 set to the authentication algorithm the resolver plans to use. 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 [RFC 1750] to derive the keying material to be used. 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 [RFC 2845] 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 assigned 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 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 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 maximum 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.

4.5 Query for Resolver Assigned Keying

Optionally, a server can accept resolver assigned keys. 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 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 mode 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.

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 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. 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 Diffie-Hellman 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 under the public key in an accompanying KEY RR [RFC 2535]. This KEY RR MUST be 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 less 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 [RFC 2437]. (Note, 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 [RFC 1750] padding.

7. IANA Considerations

This section is to be interpreted as provided in [RFC 2434].

Mode field values 0x0000 and 0xFFFF are reserved.

Mode field values 0x0001 through 0x00FF, and 0XFF00 through 0XFFFE can only be assigned by an IETF Standards Action.

Mode field values 0x0100 through 0x0FFF and 0xF0000 through 0xFEFF are allocated by IESG approval or IETF consensus.

Mode field values 0x1000 through 0xEFFF are allocated based on Specification Required as defined in [RFC 2434].

Mode values should not be changed when the status of their use changes. For example, a mode value assigned based just on providing a specification should not be changed later just because that use's status is changed to standards track.

The following assignments are documented herein:

RR Type 249 for TKEY.

TKEY Modes 1 through 5 as listed in section 2.5.

Extended RCODE Error values of 19, 20, and 21 as listed in section 2.6.

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 [RFC 2845].

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

References

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

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

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

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

[RFC 1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, September 1996.

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

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

[RFC 2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

[RFC 2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC 2437] Kaliski, B. and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0", RFC 2437, October 1998.

[RFC 2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC 2539] Eastlake, D., "Storage of Diffie-Hellman Keys in the Domain Name System (DNS)", RFC 2539, March 1999.

[RFC 2845] Vixie, P., Gudmundsson, O., Eastlake, D. and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC 2931] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

Author's Address

Donald E. Eastlake 3rd
Motorola
140 Forest Avenue
Hudson, MA 01749 USA

Phone: +1 978-562-2827 (h)
+1 508-261-5434 (w)
Fax: +1 508-261-4447 (w)
EMail: Donald.Eastlake@motorola.com

Full Copyright Statement

Acknowledgement

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