Internet Engineering Task Force
Internet-Draft
Obsoletes: 2845, 4635 (if approved)
Intended status: Standards Track
Expires: May 23, 2019

F. Dupont
S. Morris
ISC
P. Vixie
Farsight
D. Eastlake 3rd
Huawei
O. Gudmundsson
CloudFlare
B. Wellington
Akamai
November 19, 2018

Secret Key Transaction Authentication for DNS (TSIG)
draft-ietf-dnsop-rfc2845bis-02

Abstract

This protocol allows for transaction level authentication using shared secrets and one way hashing. It can be used to authenticate dynamic updates as coming from an approved client, or to authenticate responses as coming from an approved name server.

No provision has been made here for distributing the shared secrets.

This document includes revised original TSIG specifications (RFC2845) and its extension for HMAC-SHA (RFC4635).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) [RFC1034], [RFC1035] is a replicated hierarchical distributed database system that provides information fundamental to Internet operations, such as name <=> address translation and mail handling information.

In 2017, security problems in two nameservers strictly following [RFC2845] and [RFC4635] (i.e., TSIG and its HMAC-SHA extension) specifications were discovered. The implementations were fixed but, to avoid similar problems in the future, the two documents were updated and merged, producing this revised specification for TSIG.

This document specifies use of a message authentication code (MAC), either HMAC-MD5 or HMAC-SHA (keyed hash functions), to provide an efficient means of point-to-point authentication and integrity checking for DNS transactions.

The second area where the secret key based MACs specified in this document can be used is to authenticate DNS update requests as well

as transaction responses, providing a lightweight alternative to the protocol described by [RFC3007].

A further use of this mechanism is to protect zone transfers. In this case the data covered would be the whole zone transfer including any glue records sent. The protocol described by DNSSEC does not protect glue records and unsigned records unless SIG(0) (transaction signature) is used.

The authentication mechanism proposed in this document uses shared secret keys to establish a trust relationship between two entities. Such keys must be protected in a fashion similar to private keys, lest a third party masquerade as one of the intended parties (by forging the MAC). There is an urgent need to provide simple and efficient authentication between clients and local servers and this proposal addresses that need. The proposal is unsuitable for general server to server authentication for servers which speak with many other servers, since key management would become unwieldy with the number of shared keys going up quadratically. But it is suitable for many resolvers on hosts that only talk to a few recursive servers.

A server acting as an indirect caching resolver -- a "forwarder" in common usage -- might use transaction-based authentication when communicating with its small number of preconfigured "upstream" servers. Other uses of DNS secret key authentication and possible systems for automatic secret key distribution may be proposed in separate future documents.

Note that use of TSIG presumes prior agreement between the two parties involved (e.g., resolver and server) as to any algorithm and key to be used.

Since the publication of first version of this document ([RFC2845]) a mechanism based on asymmetric signatures using the SIG RR was specified (SIG(0) [RFC2931]) whereas this document uses symmetric authentication codes calculated by HMAC [RFC2104] using strong hash functions.

2. Key words

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.

3. New Assigned Numbers

RRTYPE = TSIG (250)
ERROR = 0..15 (a DNS RCODE)
ERROR = 16 (BADSIG)
ERROR = 17 (BADKEY)
ERROR = 18 (BADTIME)
ERROR = 22 (BADTRUNC)

(See [RFC6895] Section 2.3 concerning the assignment of the value 16 to BADSIG.)

4. TSIG RR Format

4.1. TSIG RR Type

To provide secret key authentication, we use a new RR type whose mnemonic is TSIG and whose type code is 250. TSIG is a meta-RR and MUST NOT be cached. TSIG RRs are used for authentication between DNS entities that have established a shared secret key. TSIG RRs are dynamically computed to cover a particular DNS transaction and are not DNS RRs in the usual sense.

4.2. TSIG Calculation

As the TSIG RRs are related to one DNS request/response, there is no value in storing or retransmitting them, thus the TSIG RR is discarded once it has been used to authenticate a DNS message. Recommendations concerning the message digest algorithm can be found in Section 7. All multi-octet integers in the TSIG record are sent in network byte order (see [RFC1035] 2.3.2).

4.3. TSIG Record Format

NAME The name of the key used in domain name syntax. The name should reflect the names of the hosts and uniquely identify the key among a set of keys these two hosts may share at any given time. If hosts A.site.example and B.example.net share a key, possibilities for the key name include <id>.A.site.example, <id>.B.example.net, and <id>.A.site.example.B.example.net. It should be possible for more than one key to be in simultaneous use among a set of interacting hosts. The name only needs to be meaningful to the communicating hosts but a meaningful mnemonic name as above is strongly recommended.

The name may be used as a local index to the key involved and it is recommended that it be globally unique. Where a key is just shared between two hosts, its name actually need only be

meaningful to them but it is recommended that the key name be mnemonic and incorporate the resolver and server host names in that order.

TYPE TSIG (250: Transaction SIGnature)

CLASS ANY

TTL   0

RdLen (variable)

RDATA The RDATA for a TSIG RR consists of an octet stream Algorithm Name field, a uint48_t Time Signed field, a uint16_t Fudge field, a uint16_t MAC Size field, a octet stream MAC field, a uint16_t Original ID, a uint16_t Error field, a uint16_t Other Len field and an octet stream of Other Data.

```
                            1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       /                         Algorithm Name                        /
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       |          Time Signed          +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                               |            Fudge              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |          MAC Size             |                               /
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+             MAC               /
       /                                                               /
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |          Original ID          |            Error              |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |          Other Len            |                               /
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+           Other Data          /
       /                                                               /
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The contents of the RDATA fields are:

- Algorithm Name - identifies the TSIG algorithm name in the domain name syntax.

- Time Signed - time signed as seconds since 00:00 on 1970-01-01 UTC ignoring leap seconds.

- Fudge - specifies the allowed time difference in seconds permitted in the Time Signed field.

* MAC Size - the length of MAC field in octets. Truncation is indicated by a MAC size less than the size of the keyed hash produced by the algorithm specified by the Algorithm Name.

* MAC - the contents of this field are defined by the TSIG algorithm used, possibly truncated as specified by MAC Size.

* Error - contains the expanded RCODE covering TSIG processing.

* Other Len - specifies the length of the "Other Data" field in octets.

* Other Data - this field will be empty unless the content of the Error field is BADTIME, in which case it will contain the server's current time (see Section 6.5.3).

4.4. Example

NAME	HOST.EXAMPLE.
TYPE	TSIG
CLASS	ANY
TTL	0
RdLen	As appropriate
RDATA	

Field Name	Contents
Algorithm Name	SAMPLE-ALG.EXAMPLE.
Time Signed	853804800
Fudge	300
MAC Size	As appropriate
MAC	As appropriate
Original ID	As appropriate
Error	0 (NOERROR)
Other Len	0
Other Data	Empty

5. Protocol Operation

5.1. Effects of adding TSIG to outgoing message

Once the outgoing message has been constructed, the HMAC computation can be performed. The resulting MAC will then be stored in a TSIG which is appended to the additional data section (the ARCOUNT is incremented to reflect this). If the TSIG record cannot be added without causing the message to be truncated, the server MUST alter the response so that a TSIG can be included. This response consists of only the question and a TSIG record, and has the TC bit set and RCODE 0 (NOERROR). The client SHOULD at this point retry the request using TCP (per [RFC1035] 4.2.2).

5.2. TSIG processing on incoming messages

If an incoming message contains a TSIG record, it MUST be the last record in the additional section. Multiple TSIG records are not allowed. If a TSIG record is present in any other position, the DNS message is dropped and a response with RCODE 1 (FORMERR) MUST be returned. Upon receipt of a message with exactly one correctly placed TSIG RR, the TSIG RR is copied to a safe location, removed from the DNS Message, and decremented out of the DNS message header's ARCOUNT. At this point the keyed hash (HMAC) computation is performed.

If the algorithm name or key name is unknown to the recipient, or if the MACs do not match, the whole DNS message MUST be discarded. If the message is a query, a response with RCODE 9 (NOTAUTH) MUST be sent back to the originator with TSIG ERROR 17 (BADKEY) or TSIG ERROR 16 (BADSIG). If no key is available to sign this message it MUST be sent unsigned (MAC size == 0 and empty MAC). Algorithm name and time signed and fudge fields SHOULD be copied to the response to provide off path spoof protection. A message to the system operations log SHOULD be generated, to warn the operations staff of a possible security incident in progress. Care should be taken to ensure that logging of this type of event does not open the system to a denial of service attack.

Until these error checks are successfully passed, concluding that the signature is valid, the signature MUST be considered to be invalid.

5.3. Time values used in TSIG calculations

The data digested includes the two timer values in the TSIG header in order to defend against replay attacks. If this were not done, an attacker could replay old messages but update the "Time Signed" and "Fudge" fields to make the message look new. This data is named "TSIG Timers", and for the purpose of MAC calculation they are hashed

in their "on the wire" format, in the following order: first Time Signed, then Fudge. For example:

Field Name	Value	Wire Format	Meaning
Time Signed	853804800	00 00 32 e4 07 00	Tue Jan 21 00:00:00 1997
Fudge	300	01 2C	5 minutes

5.4. TSIG Variables and Coverage

When generating or verifying the contents of a TSIG record, the following data are passed as input to MAC computation, in network byte order or wire format, as appropriate:

5.4.1. DNS Message

A whole and complete DNS message in wire format, before the TSIG RR has been added to the additional data section and before the DNS Message Header's ARCOUNT field has been incremented to contain the TSIG RR. If the message ID differs from the original message ID, the original message ID is substituted for the message ID. This could happen when forwarding a dynamic update request, for example.

5.4.2. TSIG Variables

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	(Always ANY in the current specification)
TSIG RR	TTL	(Always 0 in the current specification)
TSIG RDATA	Algorithm Name	in canonical wire format
TSIG RDATA	Time Signed	in network byte order
TSIG RDATA	Fudge	in network byte order
TSIG RDATA	Error	in network byte order
TSIG RDATA	Other Len	in network byte order
TSIG RDATA	Other Data	exactly as transmitted

The RR RDLEN and RDATA MAC Length are not included in the input to MAC computation since they are not guaranteed to be knowable before the MAC is generated.

The Original ID field is not included in this section, as it has already been substituted for the message ID in the DNS header and hashed.

For each label type, there must be a defined "Canonical wire format" that specifies how to express a label in an unambiguous way. For label type 00, this is defined in [RFC4034], for label type 01, this

is defined in [RFC6891]. The use of label types other than 00 and 01 is not defined for this specification.

5.4.3. Request MAC

When generating the MAC to be included in a response, the validated request MAC MUST be included in the MAC computation. If the request MAC failed to validate, an unsigned error message MUST be returned instead. (Section 6.3).

The request's MAC is digested in wire format, including the following fields:

Field	Type	Description
MAC Length	uint16_t	in network byte order
MAC Data	octet stream	exactly as transmitted

5.5. Component Padding

Digested components (i.e., inputs to keyed hash computation) are fed into the hashing function as a continuous octet stream with no interfield separator or padding.

6. Protocol Details

6.1. TSIG generation on requests

Client performs the keyed hash (HMAC) computation and appends a TSIG record to the additional data section and transmits the request to the server. The client MUST store the MAC from the request while awaiting an answer. The digest components for a request are:

DNS Message (request)
TSIG Variables (request)

Note that some older name servers will not accept requests with a nonempty additional data section. Clients SHOULD only attempt signed transactions with servers who are known to support TSIG and share some algorithm and secret key with the client -- so, this is not a problem in practice.

6.2. TSIG on Answers

When a server has generated a response to a signed request, it signs the response using the same algorithm and key. The server MUST NOT generate a signed response to a request if either the KEY is invalid or the MAC fails validation. It also MUST NOT not generate a signed

response to an unsigned request, except in the case of a response to a client's unsigned TKEY request if the secret key is established on the server side after the server processed the client's request. Signing responses to unsigned TKEY requests MUST be explicitly specified in the description of an individual secret key establishment algorithm [RFC3645].

The digest components are:

Request MAC
DNS Message (response)
TSIG Variables (response)

6.3. TSIG on TSIG Error returns

When a server detects an error relating to the key or MAC, the server SHOULD send back an unsigned error message (MAC size == 0 and empty MAC). It MUST NOT send back a signed error message.

If an error is detected relating to the TSIG validity period or the MAC is too short for the local policy, the server SHOULD send back a signed error message. The digest components are:

Request MAC (if the request MAC validated)
DNS Message (response)
TSIG Variables (response)

The reason that the request is not included in this MAC in some cases is to make it possible for the client to verify the error. If the error is not a TSIG error the response MUST be generated as specified in Section 6.2.

6.4. TSIG on zone transfer over a TCP connection

A zone transfer over a DNS TCP session can include multiple DNS messages. Using TSIG on such a connection can protect the connection from hijacking and provide data integrity. The TSIG MUST be included on the first and last DNS messages, and SHOULD be placed on all intermediary messages. For backward compatibility, a client which receives DNS messages and verifies TSIG MUST accept up to 99 intermediary messages without a TSIG. The first envelope is processed as a standard answer, and subsequent messages have the following digest components:

Prior MAC (running)
DNS Messages (any unsigned messages since the last TSIG)
TSIG Timers (current message)

This allows the client to rapidly detect when the session has been altered; at which point it can close the connection and retry. If a client TSIG verification fails, the client MUST close the connection. If the client does not receive TSIG records frequently enough (as specified above) it SHOULD assume the connection has been hijacked and it SHOULD close the connection. The client SHOULD treat this the same way as they would any other interrupted transfer (although the exact behavior is not specified).

6.5. Server TSIG checks

Upon receipt of a message, server will check if there is a TSIG RR. If one exists, the server is REQUIRED to return a TSIG RR in the response. The server MUST perform the following checks in the following order, check KEY, check MAC, check TIME values, check Truncation policy.

6.5.1. Key check and error handling

If a non-forwarding server does not recognize the key used by the client, the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 17 (BADKEY). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error. (Special considerations apply to forwarding servers, see Section 6.7.)

6.5.2. MAC check and error handling

If a TSIG fails to verify, the server MUST generate an error response as specified in Section 6.3 with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response MUST be unsigned as specified in Section 6.3. The server SHOULD log the error.

6.5.2.1. Specifying Truncation

When space is at a premium and the strength of the full length of a MAC is not needed, it is reasonable to truncate the keyed hash and use the truncated value for authentication. HMAC SHA-1 truncated to 96 bits is an option available in several IETF protocols, including IPsec and TLS.

Processing of a truncated MAC follows these rules

1. If "MAC size" field is greater than keyed hash output length:

   This case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.

2. If "MAC size" field equals keyed hash output length:

   The entire output keyed hash output is present and used.

3. "MAC size" field is less than keyed hash output length but greater than that specified in case 4, below:

   This is sent when the signer has truncated the keyed hash output to an allowable length, as described in [RFC2104], taking initial octets and discarding trailing octets. TSIG truncation can only be to an integral number of octets. On receipt of a DNS message with truncation thus indicated, the locally calculated MAC is similarly truncated and only the truncated values are compared for authentication. The request MAC used when calculating the TSIG MAC for a reply is the truncated request MAC.

4. "MAC size" field is less than the larger of 10 (octets) and half the length of the hash function in use:

   With the exception of certain TSIG error messages described in Section 6.3, where it is permitted that the MAC size be zero, this case MUST NOT be generated and, if received, MUST cause the DNS message to be dropped and RCODE 1 (FORMERR) to be returned.

6.5.3. Time check and error handling

If the server time is outside the time interval specified by the request (which is: Time Signed, plus/minus Fudge), the server MUST generate an error response with RCODE 9 (NOTAUTH) and TSIG ERROR 18 (BADTIME). The server SHOULD also cache the most recent time signed value in a message generated by a key, and SHOULD return BADTIME if a message received later has an earlier time signed value. A response indicating a BADTIME error MUST be signed by the same key as the request. It MUST include the client's current time in the time signed field, the server's current time (a uint48_t) in the other data field, and 6 in the other data length field. This is done so that the client can verify a message with a BADTIME error without the verification failing due to another BADTIME error. The data signed is specified in Section 6.3. The server SHOULD log the error.

6.5.4. Truncation check and error handling

If a TSIG is received with truncation that is permitted under Section 6.5.2.1 above but the MAC is too short for the local policy in force, an RCODE 9 (NOTAUTH) and TSIG ERROR 22 (BADTRUNC) MUST be returned. The server SHOULD log the error.

6.6. Client processing of answer

When a client receives a response from a server and expects to see a TSIG, it first checks if the TSIG RR is present in the response. Otherwise, the response is treated as having a format error and discarded. The client then extracts the TSIG, adjusts the ARCOUNT, and calculates the MAC in the same way as the server, applying the same rules to decide if truncated MAC is valid. If the TSIG does not validate, that response MUST be discarded, unless the RCODE is 9 (NOTAUTH), in which case the client SHOULD attempt to verify the response as if it were a TSIG Error response, as specified in Section 6.3. A message containing an unsigned TSIG record or a TSIG record which fails verification SHOULD NOT be considered an acceptable response; the client SHOULD log an error and continue to wait for a signed response until the request times out.

6.6.1. Key error handling

If an RCODE on a response is 9 (NOTAUTH), and the response TSIG validates, and the TSIG key is different from the key used on the request, then this is a Key error. The client MAY retry the request using the key specified by the server. This should never occur, as a server MUST NOT sign a response with a different key than signed the request.

6.6.2. MAC error handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC error, and client MAY retry the request with a new request ID but it would be better to try a different shared key if one is available. Clients SHOULD keep track of how many MAC errors are associated with each key. Clients SHOULD log this event.

6.6.3. Time error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 18 (BADTIME), or the current time does not fall in the range specified in the TSIG record, then this is a Time error. This is an indication that the client and server clocks are not synchronized. In this case the client SHOULD log the event. DNS resolvers MUST NOT adjust any clocks in the client based on BADTIME errors, but the server's time in the other data field SHOULD be logged.

6.6.4. Truncation error handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 22 (BADTRUNC) then this is a Truncation error. The client MAY retry with a lesser truncation up to the full HMAC output (no truncation),

using the truncation used in the response as a hint for what the server policy allowed (Section 8). Clients SHOULD log this event.

6.7. Special considerations for forwarding servers

A server acting as a forwarding server of a DNS message SHOULD check for the existence of a TSIG record. If the name on the TSIG is not of a secret that the server shares with the originator the server MUST forward the message unchanged including the TSIG. If the name of the TSIG is of a key this server shares with the originator, it MUST process the TSIG. If the TSIG passes all checks, the forwarding server MUST, if possible, include a TSIG of his own, to the destination or the next forwarder. If no transaction security is available to the destination and the response has the AD flag (see [RFC4035]), the forwarder MUST unset the AD flag before adding the TSIG to the answer.

7. Algorithms and Identifiers

The only message digest algorithm specified in the first version of these specifications [RFC2845] was "HMAC-MD5" (see [RFC1321], [RFC2104]). The "HMAC-MD5" algorithm is mandatory to implement for interoperability.

The use of SHA-1 [FIPS180-4], [RFC3174], (which is a 160-bit hash as compared to the 128 bits for MD5), and additional hash algorithms in the SHA family [FIPS180-4], [RFC3874], [RFC6234] with 224, 256, 384, and 512 bits may be preferred in some cases. This is because increasingly successful cryptanalytic attacks are being made on the shorter hashes.

Use of TSIG between two DNS agents is by mutual agreement. That agreement can include the support of additional algorithms and criteria as to which algorithms and truncations are acceptable, subject to the restriction and guidelines in Section 6.5.2.1 above. Key agreement can be by the TKEY mechanism [RFC2930] or some other mutually agreeable method.

The current HMAC-MD5.SIG-ALG.REG.INT and gss-tsig [RFC3645] identifiers are included in the table below for convenience. Implementations that support TSIG MUST also implement HMAC SHA1 and HMAC SHA256 and MAY implement gss-tsig and the other algorithms listed below.

Requirement	Name
Mandatory	HMAC-MD5.SIG-ALG.REG.INT
Optional	gss-tsig
Mandatory	hmac-sha1
Optional	hmac-sha224
Mandatory	hmac-sha256
Optional	hmac-sha384
Optional	hmac-sha512

SHA-1 truncated to 96 bits (12 octets) SHOULD be implemented.

8. TSIG Truncation Policy

As noted above, two DNS agents (e.g., resolver and server) must mutually agree to use TSIG. Implicit in such an "agreement" are criteria as to acceptable keys and algorithms and, with the extensions in this document, truncations. Note that it is common for implementations to bind the TSIG secret key or keys that may be in place at two parties to particular algorithms. Thus, such implementations only permit the use of an algorithm if there is an associated key in place. Receipt of an unknown, unimplemented, or disabled algorithm typically results in a BADKEY error.

Local policies MAY require the rejection of TSIGs, even though they use an algorithm for which implementation is mandatory.

When a local policy permits acceptance of a TSIG with a particular algorithm and a particular non-zero amount of truncation, it SHOULD also permit the use of that algorithm with lesser truncation (a longer MAC) up to the full keyed hash output.

Regardless of a lower acceptable truncated MAC length specified by local policy, a reply SHOULD be sent with a MAC at least as long as that in the corresponding request. Note if the request specified a MAC length longer than the keyed hash output it will be rejected by processing rules Section 6.5.2.1 case 1.

Implementations permitting multiple acceptable algorithms and/or truncations SHOULD permit this list to be ordered by presumed strength and SHOULD allow different truncations for the same algorithm to be treated as separate entities in this list. When so implemented, policies SHOULD accept a presumed stronger algorithm and truncation than the minimum strength required by the policy.

9. Shared Secrets

Secret keys are very sensitive information and all available 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 multi-user machines, great care should be taken that unprivileged users have no access to keying material. Resolvers often run unprivileged, which means all users of a host would be able to see whatever configuration data is used by the resolver.

A name server usually runs privileged, which means its configuration data need not be visible to all users of the host. For this reason, a host that implements transaction-based authentication should probably be configured with a "stub resolver" and a local caching and forwarding name server. This presents a special problem for [RFC2136] which otherwise depends on clients to communicate only with a zone's authoritative name servers.

Use of strong random shared secrets is essential to the security of TSIG. See [RFC4086] for a discussion of this issue. The secret SHOULD be at least as long as the keyed hash output, i.e., 16 bytes for HMAC-MD5 or 20 bytes for HMAC-SHA1.

10. IANA Considerations

IANA maintains a registry of algorithm names to be used as "Algorithm Names" as defined in Section 4.3. Algorithm names are text strings encoded using the syntax of a domain name. There is no structure required other than names for different algorithms must be unique when compared as DNS names, i.e., comparison is case insensitive. Previous specifications [RFC2845] and [RFC4635] defined values for HMAC MD5 and SHA. IANA has also registered "gss-tsig" as an identifier for TSIG authentication where the cryptographic operations are delegated to the Generic Security Service (GSS) [RFC3645].

New algorithms are assigned using the IETF Consensus policy defined in [RFC8126]. The algorithm name HMAC-MD5.SIG-ALG.REG.INT looks like a fully-qualified domain name for historical reasons; other algorithm names are simple (i.e., single-component) names.

IANA maintains a registry of RCODES (error codes), including "TSIG Error values" to be used for "Error" values as defined in Section 4.3. New error codes are assigned and specified as in [RFC6895].

11. Security Considerations

The approach specified here is computationally much less expensive than the signatures specified in DNSSEC. As long as the shared secret key is not compromised, strong authentication is provided between two DNS systems, e.g., for the last hop from a local name server to the user resolver, or between primary and secondary nameservers..

Secret keys should be changed periodically. If the client host has been compromised, the server should suspend the use of all secrets known to that client. If possible, secrets should be stored in encrypted form. Secrets should never be transmitted in the clear over any network. This document does not address the issue on how to distribute secrets except that it mentions the possibilities of manual configuration and the use of TKEY [RFC2930]. Secrets SHOULD NOT be shared by more than two entities.

This mechanism does not authenticate source data, only its transmission between two parties who share some secret. The original source data can come from a compromised zone master or can be corrupted during transit from an authentic zone master to some "caching forwarder." However, if the server is faithfully performing the full DNSSEC security checks, then only security checked data will be available to the client.

A fudge value that is too large may leave the server open to replay attacks. A fudge value that is too small may cause failures if machines are not time synchronized or there are unexpected network delays. The recommended value in most situations is 300 seconds.

For all of the message authentication code algorithms listed in this document, those producing longer values are believed to be stronger; however, while there have been some arguments that mild truncation can strengthen a MAC by reducing the information available to an attacker, excessive truncation clearly weakens authentication by reducing the number of bits an attacker has to try to break the authentication by brute force [RFC2104].

Significant progress has been made recently in cryptanalysis of hash functions of the types used here, all of which ultimately derive from the design of MD4. While the results so far should not effect HMAC, the stronger SHA-1 and SHA-256 algorithms are being made mandatory due to caution. Note that today SHA-3 [FIPS202] is available as an alternative to SHA-2 using a very different design.

See also the Security Considerations section of [RFC2104] from which the limits on truncation in this RFC were taken.

11.1. Issue fixed in this document

When signing a DNS reply message using TSIG, the MAC computation uses the request message's MAC as an input to cryptographically relate the reply to the request. The original TSIG specification [RFC2845] required that the TIME values be checked before the request's MAC. If the TIME was invalid, some implementations failed to carry out further checks and could use an invalid request MAC in the signed reply.

This document proposes the principle that the request MAC must be considered to be invalid until it has been validated: until then, any answer must be unsigned. For this reason, the request MAC is now checked before the TIME value.

11.2. Why not DNSSEC?

This section from the original document [RFC2845] analyzes DNSSEC in order to justify the introduction of TSIG.

DNS has recently been extended by DNSSEC ([RFC4033], [RFC4034] and [RFC4035]) to provide for data origin authentication, and public key distribution, all based on public key cryptography and public key based digital signatures. To be practical, this form of security generally requires extensive local caching of keys and tracing of authentication through multiple keys and signatures to a pre-trusted locally configured key.

One difficulty with the DNSSEC scheme is that common DNS implementations include simple "stub" resolvers which do not have caches. Such resolvers typically rely on a caching DNS server on another host. It is impractical for these stub resolvers to perform general DNSSEC authentication and they would naturally depend on their caching DNS server to perform such services for them. To do so securely requires secure communication of queries and responses. DNSSEC provides public key transaction signatures to support this, but such signatures are very expensive computationally to generate. In general, these require the same complex public key logic that is impractical for stubs.

A second area where use of straight DNSSEC public key based mechanisms may be impractical is authenticating dynamic update [RFC2136] requests. DNSSEC provides for request signatures but with DNSSEC they, like transaction signatures, require computationally expensive public key cryptography and complex authentication logic. Secure Domain Name System Dynamic Update ([RFC3007]) describes how different keys are used in dynamically updated zones.

12. References

12.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015.

[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>.

[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>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC4635] Eastlake 3rd, D., "HMAC SHA (Hashed Message Authentication Code, Secure Hash Algorithm) TSIG Algorithm Identifiers", RFC 4635, DOI 10.17487/RFC4635, August 2006, <https://www.rfc-editor.org/info/rfc4635>.

[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>.

12.2. Informative References

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard", FIPS PUB 202, August 2015.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[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>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[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>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[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>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[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>.

[RFC3874] Housley, R., "A 224-bit One-way Hash Function: SHA-224", RFC 3874, DOI 10.17487/RFC3874, September 2004, <https://www.rfc-editor.org/info/rfc3874>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[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>.

[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>.

[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>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[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>.

Appendix A. Acknowledgments

This document consolidates and updates the earlier documents by the authors of [RFC2845] (Paul Vixie, Olafur Gudmundsson, Donald E. Eastlake 3rd and Brian Wellington) and [RFC4635] (Donald E. Eastlake 3rd). It would not be possible without their original work.

The security problem addressed by this document was reported by Clement Berthaux from Synacktiv.

Note for the RFC Editor (to be removed before publication): the first 'e' in Clement is a fact a small 'e' with acute, unicode code U+00E9. I do not know if xml2rfc supports non ASCII characters so I prefer to not experiment with it. BTW I am French too so I can help if you have questions like correct spelling...

Peter van Dijk, Benno Overeinder, Willem Toroop, Ondrej Sury, Mukund Sivaraman and Ralph Dolmans participated in the discussions that prompted this document.

Appendix B. Change History (to be removed before publication)

draft-dupont-dnsop-rfc2845bis-00

[RFC4635] was merged.

Authors of original documents were moved to Acknowledgments (Appendix A).

Section 2 was updated to [RFC8174] style.

Spit references into normative and informative references and updated them.

Added a text explaining why this document was written in the Abstract and at the beginning of the introduction.

Clarified the layout of TSIG RDATA.

Moved the text about using DNSSEC from the Introduction to the end of Security Considerations.

Added the security clarifications:

1. Emphasized that MAC is invalid until it is successfully validated.

2. Added requirement that a request MAC that has not been successfully validated MUST NOT be included into a response.

3. Added requirement that a request that has not been validated MUST NOT generate a signed response.

4. Added note about MAC too short for the local policy to Section 6.3.

5. Changed the order of server checks and swapped corresponding sections.

6. Removed the truncation size limit "also case" as it does not apply and added confusion.

7. Relocated the error provision for TSIG truncation to the new Section 6.5.4. Moved from RCODE 22 to RCODE 9 and TSIG ERROR 22, i.e., aligned with other TSIG error cases.

8. Added Section 6.6.4 about truncation error handling by clients.

9. Removed the limit to HMAC output in replies as a request which specified a MAC length longer than the HMAC output is invalid according to the first processing rule in Section 6.5.2.1.

10. Promoted the requirement that a secret length should be at least as long as the HMAC output to a SHOULD [RFC2119] key word.

11. Added a short text to explain the security issue.

draft-dupont-dnsop-rfc2845bis-01

Improved wording (post-publication comments).

Specialized and renamed the "TSIG on TCP connection" (Section 6.4) to "TSIG on zone transfer over a TCP connection". Added a SHOULD for a TSIG in each message (was envelope) for new implementations.

draft-ietf-dnsop-rfc2845bis-00

Adopted by the IETF DNSOP working group: title updated and version counter reset to 00.

draft-ietf-dnsop-rfc2845bis-01

Relationship between protocol change and principle of assuming the request MAC is invalid until validated clarified. (Jinmei Tatuya)

Cross reference to considerations for forwarding servers added. (Bob Harold)

Added text from [RFC3645] concerning the signing behavior if a secret key is added during a multi-message exchange.

Added reference to [RFC6895].

Many improvements in the wording.

Added RFC 2845 authors as co-authors of this document.

draft-ietf-dnsop-rfc2845bis-02

Added a recommendation to copy time fields in BADKEY errors. (Mark Andrews)

Authors' Addresses

Francis Dupont
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States of America

Email: Francis.Dupont@fdupont.fr

Stephen Morris
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
United States of America

Email: stephen@isc.org

Paul Vixie
Farsight Security Inc
177 Bovet Road, Suite 180
San Mateo, CA 94402
United States of America

Email: paul@redbarn.org

Donald E. Eastlake 3rd
Huawei Technologies
155 Beaver Street
Milford, MA 01753
United States of America

Email: d3e3e3@gmail.com

Olafur Gudmundsson
CloudFlare
San Francisco, CA 94107
United States of America

Email: olafur+ietf@cloudflare.com

Brian Wellington
Akamai
United States of America

Email: bwelling@akamai.com