Stream: Internet Engineering Task Force (IETF)
RFC: 8945
STD: 93
Obsoletes: 2845, 4635
Category: Standards Track
Published: November 2020
ISSN: 2070-1721
Authors:
F. Dupont, *ISC*
S. Morris, *Unaffiliated*
P. Vixie, *Farsight*
D. Eastlake 3rd, *Futurewei*
O. Gudmundsson, *Cloudflare*
B. Wellington, *Akamai*

RFC 8945
Secret Key Transaction Authentication for DNS (TSIG)

Abstract

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

No recommendation is made here for distributing the shared secrets; it is expected that a network administrator will statically configure name servers and clients using some out-of-band mechanism.

This document obsoletes RFCs 2845 and 4635.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8945.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

1.1. Background

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

This document specifies use of a message authentication code (MAC), generated using certain keyed hash functions, to provide an efficient means of point-to-point authentication and integrity checking for DNS transactions. Such transactions include DNS update requests and responses for which this can provide a lightweight alternative to the secure DNS dynamic update 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 ([RFC4033], [RFC4034], [RFC4035]) does not protect glue records and unsigned records.

The authentication mechanism proposed here provides a simple and efficient authentication between clients and servers, by using shared secret keys to establish a trust relationship between two entities. Such keys must be protected in a manner similar to private keys, lest a third party masquerade as one of the intended parties (by forging the MAC). The proposal is unsuitable for general server-to-server authentication and for servers that 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.

1.2. Protocol Overview

Secret Key Transaction Authentication makes use of signatures on messages sent between the parties involved (e.g., resolver and server). These are known as "transaction signatures", or TSIG. For historical reasons, in this document, they are referred to as message authentication codes (MACs).

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. The way that this agreement is reached is outside the scope of the document.

A DNS message exchange involves the sending of a query and the receipt of one of more DNS messages in response. For the query, the MAC is calculated based on the hash of the contents and the agreed TSIG key. The MAC for the response is similar but also includes the MAC of the query as part of the calculation. Where a response comprises multiple packets, the calculation of the MAC associated with the second and subsequent packets includes in its inputs the MAC for the preceding packet. In this way, it is possible to detect any interruption in the packet sequence, although not its premature termination.

The MAC is contained in a TSIG resource record included in the additional section of the DNS message.

1.3. Document History

TSIG was originally specified by [RFC2845]. In 2017, two name server implementations strictly following that document (and the related [RFC4635]) were discovered to have security problems related to this feature ([CVE-2017-3142], [CVE-2017-3143], [CVE-2017-11104]). 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.

While TSIG implemented according to this RFC provides for enhanced security, there are no changes in interoperability. TSIG on the wire is still the same mechanism described in [RFC2845]; only the checking semantics have been changed. See Section 10.1 for further details.

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. Assigned Numbers

This document defines the following Resource Record (RR) type and associated value:

TSIG (250)

In addition, the document also defines the following DNS RCODEs and associated names:

16 (BADSIG)

17 (BADKEY)

18 (BADTIME)

22 (BADTRUNC)

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

These RCODES may appear within the "Error" field of a TSIG RR.

4. TSIG RR Format

4.1. TSIG RR Type

To provide secret key authentication, we use an 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.

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.

4.2. TSIG Record Format

The fields of the TSIG RR are described below. All multi-octet integers in the record are sent in network byte order (see Section 2.3.2 of [RFC1035]).

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. For example, 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. This allows for periodic key rotation as per best operational practices, as well as algorithm agility as indicated by [RFC7696].

The name may be used as a local index to the key involved, but 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 names of participating agents or resources as suggested above.

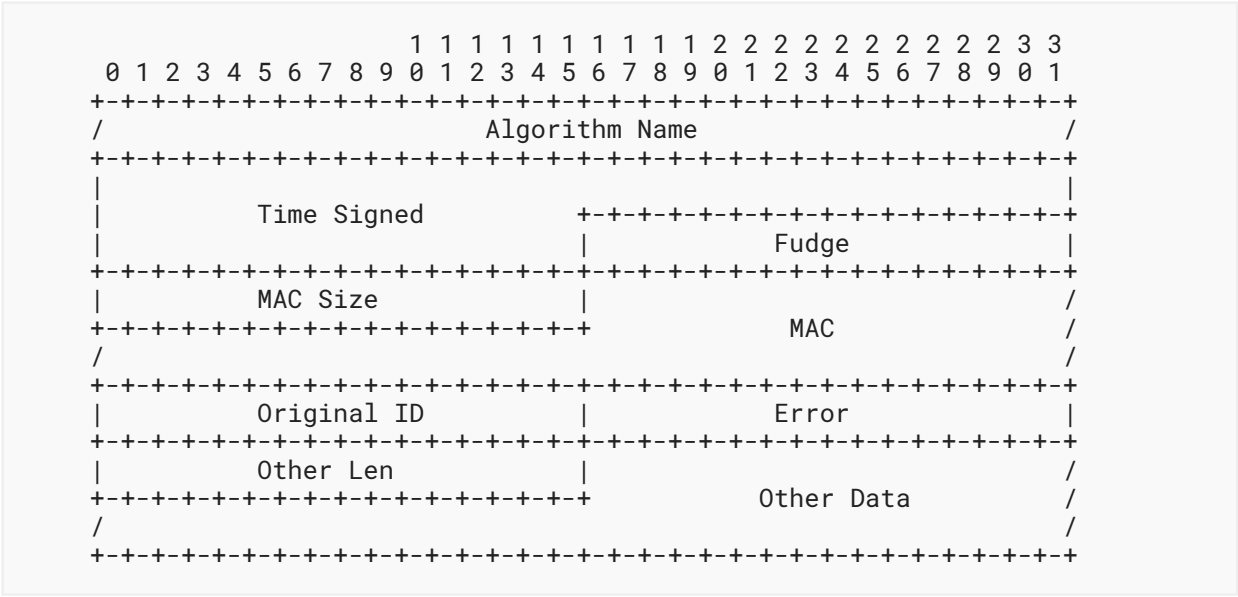
TYPE: This **MUST** be TSIG (250: Transaction SIGnature).

CLASS: This **MUST** be ANY.

TTL: This **MUST** be 0.

RDLENGTH: (variable)

RDATA: The RDATA for a TSIG RR consists of a number of fields, described below:



The contents of the RDATA fields are:

Algorithm Name:
: an octet sequence identifying the TSIG algorithm in the domain name syntax. (Allowed names are listed in Table 3.) The name is stored in the DNS name wire format as described in [RFC1034]. As per [RFC3597], this name **MUST NOT** be compressed.

Time Signed:
: an unsigned 48-bit integer containing the time the message was signed as seconds since 00:00 on 1970-01-01 UTC, ignoring leap seconds.

Fudge:
: an unsigned 16-bit integer specifying the allowed time difference in seconds permitted in the Time Signed field.

MAC Size:
: an unsigned 16-bit integer giving the length of the 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:
: a sequence of octets whose contents are defined by the TSIG algorithm used, possibly truncated as specified by the MAC Size. The length of this field is given by the MAC Size. Calculation of the MAC is detailed in Section 4.3.

Original ID:
: an unsigned 16-bit integer holding the message ID of the original request message. For a TSIG RR on a request, it is set equal to the DNS message ID. In a TSIG attached to a response -- or in cases such as the forwarding of a dynamic update request -- the field contains the ID of the original DNS request.

Error:
: in responses, an unsigned 16-bit integer containing the extended RCODE covering TSIG processing. In requests, this **MUST** be zero.

Other Len:
: an unsigned 16-bit integer specifying the length of the Other Data field in octets.

Other Data:
: additional data relevant to the TSIG record. In responses, this will be empty (i.e., Other Len will be zero) unless the content of the Error field is BADTIME, in which case it will be a 48-bit unsigned integer containing the server's current time as the number of seconds since 00:00 on 1970-01-01 UTC, ignoring leap seconds (see Section 5.2.3). This document assigns no meaning to its contents in requests.

4.3. MAC Computation

When generating or verifying the contents of a TSIG record, the data listed in the rest of this section are passed, in the order listed below, as input to MAC computation. The data are passed in network byte order or wire format, as appropriate and are fed into the hashing function as a continuous octet sequence with no interfield separator or padding.

4.3.1. Request MAC

Only included in the computation of a MAC for a response message (or the first message in a multi-message 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 5.3.2).

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

Field	Type	Description
MAC Size	Unsigned 16-bit integer	in network byte order
MAC Data	octet sequence	exactly as transmitted

Table 1: Request's MAC

Special considerations apply to the TSIG calculation for the second and subsequent messages in a response that consists of multiple DNS messages (e.g., a zone transfer). These are described in Section 5.3.1.

4.3.2. DNS Message

In the MAC computation, the whole/complete DNS message in wire format is used.

When creating an outgoing message, the TSIG is based on the message content before the TSIG RR has been added to the additional section and before the DNS Message Header's ARCOUNT has been incremented to include the TSIG RR.

When verifying an incoming message, the TSIG is checked against the message after the TSIG RR has been removed, the ARCOUNT decremented, and the message ID replaced by the original message ID from the TSIG if those IDs differ. (This could happen, for example, when forwarding a dynamic update request.)

4.3.3. TSIG Variables

Also included in the digest is certain information present in the TSIG RR. Adding this data provides further protection against an attempt to interfere with the message.

Source	Field Name	Notes
TSIG RR	NAME	Key name, in canonical wire format
TSIG RR	CLASS	**MUST** be ANY
TSIG RR	TTL	**MUST** be 0
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

Table 2: TSIG Variables

The RR RDLENGTH and RDATA MAC Size 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 Section 6.2 of [RFC4034]. The use of label types other than 00 is not defined for this specification.

4.3.3.1. 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. The two fields are collectively named "TSIG Timers", and for the purpose of MAC calculation, they are hashed in their wire format, in the following order: first Time Signed, then Fudge.

5. Protocol Details

5.1. Generation of TSIG on Requests

Once the outgoing record has been constructed, the client performs the keyed hash (Hashed Message Authentication Code (HMAC)) computation, appends a TSIG record with the calculated MAC to the additional section (incrementing the ARCOUNT to reflect the additional RR), and transmits the request to the server. This TSIG record **MUST** be the only TSIG RR in the message and **MUST** be the last record in the additional data section. The client **MUST** store the MAC and the key name from the request while awaiting an answer.

The digest components for a request are:

DNS Message (request)

TSIG Variables (request)

5.2. Server Processing of Request

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 multiple TSIG records are detected or 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, a copy of the TSIG RR is stored and the TSIG RR is removed from the DNS message and decremented out of the DNS message header's ARCOUNT.

If the TSIG RR cannot be interpreted, the server **MUST** regard the message as corrupt and return a FORMERR to the server. Otherwise, the server is **REQUIRED** to return a TSIG RR in the response.

To validate the received TSIG RR, the server **MUST** perform the following checks in the following order:

1. Check key
2. Check MAC
3. Check time values
4. Check truncation policy

5.2.1. Key Check and Error Handling

If a non-forwarding server does not recognize the key or algorithm used by the client (or recognizes the algorithm but does not implement it), 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 5.3.2. The server **SHOULD** log the error. (Special considerations apply to forwarding servers; see Section 5.5.)

5.2.2. MAC Check and Error Handling

Using the information in the TSIG, the server **MUST** verify the MAC by doing its own calculation and comparing the result with the MAC received. If the MAC fails to verify, the server **MUST** generate an error response as specified in Section 5.3.2 with RCODE 9 (NOTAUTH) and TSIG ERROR 16 (BADSIG). This response **MUST** be unsigned, as specified in Section 5.3.2. The server **SHOULD** log the error.

5.2.2.1. MAC 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. However, while this option is kept for backwards compatibility, it may not provide a security level appropriate for all cases in the modern environment. In these cases, it is preferable to use a hashing algorithm such as SHA-256-128, SHA-384-192, or SHA-512-256 [RFC4868].

Processing of a truncated MAC follows these rules:

If the MAC Size field is greater than the 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.

If the MAC Size field equals the keyed hash output length: The entire keyed hash output is present and used.

If the 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 5.3.2, 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.

Otherwise: 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.

5.2.3. Time Check and Error Handling

If the server time is outside the time interval specified by the request (which is the Time Signed value plus/minus the Fudge value), 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 (an unsigned 48-bit integer) in the Other Data field, and 6

in the Other Len 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. In addition, the Fudge field **MUST** be set to the fudge value received from the client. The data signed is specified in Section 5.3.2. The server **SHOULD** log the error.

Caching the most recent Time Signed value and rejecting requests with an earlier one could lead to valid messages being rejected if transit through the network led to UDP packets arriving in a different order to the one in which they were sent. Implementations should be aware of this possibility and be prepared to deal with it, e.g., by retransmitting the rejected request with a new TSIG once outstanding requests have completed or the time given by their Time Signed value plus the Fudge value has passed. If implementations do retry requests in these cases, a limit **SHOULD** be placed on the maximum number of retries.

5.2.4. Truncation Check and Error Handling

If a TSIG is received with truncation that is permitted per Section 5.2.2.1 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.

5.3. Generation of 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 (e.g., key name or algorithm name are unknown) or the MAC fails validation; see Section 5.3.2 for details of responding in these cases.

It also **MUST 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 used to generate a TSIG on a response are:

Request MAC

DNS Message (response)

TSIG Variables (response)

(This calculation is different for the second and subsequent message in a multi-message answer; see below.)

If addition of the TSIG record will cause the message to be truncated, the server **MUST** alter the response so that a TSIG can be included. This response contains only the question and a TSIG record, has the TC bit set, and has an RCODE of 0 (NOERROR). At this point, the client **SHOULD** retry the request using TCP (as per Section 4.2.2 of [RFC1035]).

5.3.1. TSIG on TCP Connections

A DNS TCP session, such as a zone transfer, can include multiple DNS messages. Using TSIG on such a connection can protect the connection from an attack and provide data integrity. The TSIG **MUST** be included on all DNS messages in the response. For backward compatibility, a client that receives DNS messages and verifies TSIG **MUST** accept up to 99 intermediary messages without a TSIG and **MUST** verify that both the first and last message contain a TSIG.

The first message is processed as a standard answer (see Section 5.3), but subsequent messages have the following digest components:

Prior MAC (running)

DNS Messages (any unsigned messages since the last TSIG)

TSIG Timers (current message)

The "Prior MAC" is the MAC from the TSIG attached to the last message containing a TSIG. "DNS Messages" comprises the concatenation (in message order) of all messages after the last message that included a TSIG and includes the current message. "TSIG Timers" comprises the Time Signed and Fudge fields (in that order) pertaining to the message for which the TSIG was created; this means that the successive TSIG records in the stream will have non-decreasing Time Signed values. Note that only the timers are included in the second and subsequent messages, not all the TSIG variables.

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).

5.3.2. Generation of TSIG on Error Returns

When a server detects an error relating to the key or MAC in the incoming request, 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 MAC 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 5.3.

5.4. 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. If not, the response is treated as having a format error and is discarded.

If the TSIG RR is present, the client performs the same checks as described in Section 5.2. If the TSIG RR is unsigned as specified in Section 5.3.2 or does not validate, the message **MUST** be discarded unless the RCODE is 9 (NOAUTH). In this case, the client **SHOULD** attempt to verify the response as if it were a TSIG error, as described in the following subsections.

Regardless of the RCODE, a message containing a TSIG RR that is unsigned as specified in Section 5.3.2 or that fails verification **SHOULD NOT** be considered an acceptable response, as it may have been spoofed or manipulated. Instead, the client **SHOULD** log an error and continue to wait for a signed response until the request times out.

5.4.1. Key Error Handling

If an RCODE on a response is 9 (NOTAUTH), but the response TSIG validates and the TSIG key is recognized by the client but is different from that used on the request, then this is a key-related error. The client **MAY** retry the request using the key specified by the server. However, this should never occur, as a server **MUST NOT** sign a response with a different key to that used to sign the request.

5.4.2. MAC Error Handling

If the response RCODE is 9 (NOTAUTH) and TSIG ERROR is 16 (BADSIG), this is a MAC-related error, and clients **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.

5.4.3. 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-related 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.

5.4.4. Truncation Error Handling

If the response RCODE is 9 (NOTAUTH) and the TSIG ERROR is 22 (BADTRUNC), then this is a truncation-related 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 7). Clients **SHOULD** log this event.

5.5. 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 its own to the destination or the next forwarder. If no transaction security is available to the destination and the message is a query, and if the corresponding response has the AD flag (see [RFC4035]) set, the forwarder **MUST** clear the AD flag before adding the TSIG to the response and returning the result to the system from which it received the query.

6. Algorithms and Identifiers

The only message digest algorithm specified in the first version of these specifications [RFC2845] was "HMAC-MD5" (see [RFC1321] and [RFC2104]). Although a review of its security some years ago [RFC6151] concluded that "it may not be urgent to remove HMAC-MD5 from the existing protocols", with the availability of more secure alternatives, the opportunity has been taken to make the implementation of this algorithm optional.

[RFC4635] added mandatory support in TSIG for SHA-1 [FIPS180-4] [RFC3174]. SHA-1 collisions have been demonstrated [SHA1SHAMBLES], so the MD5 security considerations described in Section 2 of [RFC6151] apply to SHA-1 in a similar manner. Although support for hmac-sha1 in TSIG is still mandatory for compatibility reasons, existing uses **SHOULD** be replaced with hmac-sha256 or other SHA-2 digest algorithms ([FIPS180-4], [RFC3874], [RFC6234]).

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 5.2.2.1. Key agreement can be by the TKEY mechanism [RFC2930] or some other mutually agreeable method.

Implementations that support TSIG **MUST** also implement HMAC SHA1 and HMAC SHA256 and **MAY** implement gss-tsig and the other algorithms listed below. SHA-1 truncated to 96 bits (12 octets) **SHOULD** be implemented.

Algorithm Name	Implementation	Use
HMAC-MD5.SIG-ALG.REG.INT	**MAY**	**MUST NOT**
gss-tsig	**MAY**	**MAY**
hmac-sha1	**MUST**	**NOT RECOMMENDED**
hmac-sha224	**MAY**	**MAY**
hmac-sha256	**MUST**	**RECOMMENDED**

Algorithm Name	Implementation	Use
hmac-sha256-128	**MAY**	**MAY**
hmac-sha384	**MAY**	**MAY**
hmac-sha384-192	**MAY**	**MAY**
hmac-sha512	**MAY**	**MAY**
hmac-sha512-256	**MAY**	**MAY**

Table 3: Algorithms for Implementations Supporting TSIG

7. 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, algorithms, and (with the extensions in this document) truncations. 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 5.2.2.1, 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.

8. 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 so 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 are 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 [RFC2104].

9. IANA Considerations

IANA maintains a registry of algorithm names to be used as "Algorithm Names", as defined in Section 4.2 [IANA-TSIG]. Algorithm names are text strings encoded using the syntax of a domain name. There is no structure to the names, and algorithm names are compared as if they were DNS names, i.e., comparison is case insensitive. Previous specifications ([RFC2845] and [RFC4635]) defined values for the HMAC-MD5 and some HMAC-SHA algorithms. 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]. This document adds to the allowed algorithms, and the registry has been updated with the names listed in Table 3.

New algorithms are assigned using the IETF Review 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, single-component names.

IANA maintains a registry of RCODEs (error codes) (see [IANA-RCODEs], including "TSIG Error values" to be used for "Error" values, as defined in Section 4.2. This document defines the RCODEs as described in Section 3. New error codes are assigned and specified as in [RFC6895].

10. 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 name servers.

Recommendations for choosing and maintaining secret keys can be found in [RFC2104]. 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 an 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; any such additional sharing would allow any party knowing the key to impersonate any other such party to members of the group.

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.

To prevent cross-algorithm attacks, there **SHOULD** only be one algorithm associated with any given key name.

In several cases where errors are detected, an unsigned error message must be returned. This can allow for an attacker to spoof or manipulate these responses. Section 5.4 recommends logging these as errors and continuing to wait for a signed response until the request times out.

Although the strength of an algorithm determines its security, there have been some arguments that mild truncation can strengthen a MAC by reducing the information available to an attacker. However, 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. While the results so far should not affect HMAC, the stronger SHA-256 algorithm is being made mandatory as a precaution.

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

10.1. 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 makes it mandatory that the request MAC is 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 values.

10.2. Why Not DNSSEC?

DNS has 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.

11. References

11.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

[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>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[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>.

11.2. Informative References

[CVE-2017-11104] Common Vulnerabilities and Exposures, "CVE-2017-11104: Improper TSIG validity period check can allow TSIG forgery", June 2017, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11104>.

[CVE-2017-3142] Common Vulnerabilities and Exposures, "CVE-2017-3142: An error in TSIG authentication can permit unauthorized zone transfers", June 2017, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3142>.

[CVE-2017-3143] Common Vulnerabilities and Exposures, "CVE-2017-3143: An error in TSIG authentication can permit unauthorized dynamic updates", June 2017, <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3143>.

[IANA-RCODEs] IANA, "DNS RCODEs", <https://www.iana.org/assignments/dns-parameters/>.

[IANA-TSIG] IANA, "TSIG Algorithm Names", <https://www.iana.org/assignments/tsig-algorithm-names/>.

[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>.

[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>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[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>.

[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>.

[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>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

[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>.

[SHA1SHAMBLES] Leurent, G. and T. Peyrin, "SHA-1 is a Shambles", January 2020, <https://eprint.iacr.org/2020/014.pdf>.

Acknowledgements

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

Peter van Dijk, Benno Overeinder, Willem Toroop, Ondrej Sury, Mukund Sivaraman, and Ralph Dolmans participated in the discussions that prompted this document. Mukund Sivaraman, Martin Hoffman, and Tony Finch made extremely helpful suggestions concerning the structure and wording of the updated document.

Stephen Morris would like to thank Internet Systems Consortium for its support of his participation in the creation of this document.

Authors' Addresses

Francis Dupont
Internet Systems Consortium, Inc.
PO Box 360
Newmarket, NH 03857
United States of America
Email: Francis.Dupont@fdupont.fr

Stephen Morris
Unaffiliated
United Kingdom
Email: sa.morris8@gmail.com

Paul Vixie
Farsight Security Inc
Suite 180
177 Bovet Road
San Mateo, CA 94402
United States of America
Email: paul@redbarn.org

Donald E. Eastlake 3rd
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703
United States of America
Email: d3e3e3@gmail.com

Olafur Gudmundsson
Cloudflare
United States of America
Email: olafur+ietf@cloudflare.com

Brian Wellington
Akamai
United States of America
Email: bwelling@akamai.com