Cryptographic Algorithms, Use, & Implementation Requirements for TCP Authentication Option
draft-lebovitz-ietf-tcpm-tcp-ao-crypto-00

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on September 4, 2009.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

The TCP Authentication Option, TCP-AO, relies on security algorithms to provide authentication between two end-points. There are many
such algorithms available, and two TCP-AO systems cannot interoperate unless they are using the same algorithm(s). This document specifies the algorithms and attributes that can be used in TCP-AO manual key mode. It also defines a UI labels framework that will be used across implementations to aid administrators in quickly achieving successful TCP-AO connections, something that will become far more important once a key management protocol (KMP) is defined for TCP-AO.

Table of Contents

1. Introduction .................................................. 3
2. Requirements .................................................. 3
   2.1. Requirements Language ...................................... 3
   2.2. Algorithm Requirements ..................................... 4
   2.3. Requirements for Future MAC Algorithms .................... 4
3. Algorithms Specified ........................................... 4
   3.1. Key Derivation Functions (KDFs) ............................ 5
      3.1.1. The Use of KDF_HMAC_SHA1 ................................ 7
      3.1.2. The Use of KDF_AES_128_CMAC ............................ 7
   3.2. MAC Algorithms ............................................. 9
      3.2.1. The Use of HMAC-SHA-1-96 ............................... 10
      3.2.2. The Use of AES-128-CMAC-96 ............................ 10
4. UI Labels ....................................................... 11
   4.1. Label "Option-A" ............................................ 12
   4.2. Label "Option-B" ............................................ 12
5. Change History (RFC Editor: Delete before publishing) .......... 13
7. Security Considerations ......................................... 13
8. IANA Considerations ............................................ 14
9. Acknowledgements ................................................ 14
10. References ..................................................... 15
    10.1. Normative References ....................................... 15
    10.2. Informative References ..................................... 15
Author's Address .................................................. 16
1. Introduction

This document is a companion to TCP-AO [TCP-AO]
[I-D.ietf-tcpm-tcp-auth-opt]. Like most security protocols, TCP-AO
allows users to chose which cryptographic algorithm(s) they want to
use to meet their security needs.

TCP-AO provides cryptographic authentication and message integrity
verification between to end-points. In order to accomplish this
function, one employs message authentication codes (MACs). There are
various ways to create MACs. The use of hashed-based MACs (HMAC) in
Internet protocols is defined in [2104]. The use of cipher-based
MACs (CMAC) in Internet protocols is defined in [RFC4493].

This RFC discusses the requirements for implementations to support
two MACs used in TCP-AO, both now and in the future, and includes the
rationale behind the present and future requirements. The document
then defines the use of those two MACs with TCP-AO. The document
then discusses "UI Labels" in implementations, why they have been
found to be so important to deployment success in other security
protocols, and specifies "UI Labels" for TCP-AO. (Note: these labels
are fairly unimportant now, but will become far more important once a
key management protocol, or KMP, is defined for use with TCP-AO).

2. Requirements

2.1. Requirements Language

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 [RFC2119].

We define some additional terms here:

SHOULD+: This term means the same as SHOULD. However, it is
likely that an algorithm marked as SHOULD+ will be
promoted at some future time to be a MUST.

SHOULD-: This term means the same as SHOULD. However, it is
likely that an algorithm marked as SHOULD- will be
deprecated to a MAY or worse in a future version of this
document.

MUST-: This term means the same as MUST. However, we expect
that at some point in the future this algorithm will no
longer be a MUST.
2.2. Algorithm Requirements

In this the first RFC specifying cryptography for TCP-AO, we specify one MAC algorithm as a MUST- and the other as a SHOULD+.

This table lists authentication algorithms for the TCP-AO protocol.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Authentication Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUST-</td>
<td>HMAC-SHA-1-96 [RFC2404]</td>
</tr>
<tr>
<td>SHOULD+</td>
<td>AES-128-CMAC-96 [RFC4493]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Key Derivation Function (KDF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUST-</td>
<td>KDF_HMAC_SHA1</td>
</tr>
<tr>
<td>SHOULD+</td>
<td>KDF_AES_128_CMAC</td>
</tr>
</tbody>
</table>

Notes: (1) The security issues driving the migration from SHA-1 to SHA-256 for digital signatures [HMAC-ATTACK] [HMAC-ATTACK] do not immediately render SHA-1 weak for this application of SHA-1 in HMAC mode. The security strength of SHA-1 HMACs should be sufficient for the foreseeable future, especially given that the tags are truncated to 96 bits. However, while it’s clear that the attacks aren’t practical on SHA-1, these types of analysis are mounting and could potentially pose a concern for HMAC forgery if they were significantly improved, over time. In anticipation of SHA-1’s growing less dependable over time, but given its wide deployment and current strength, it is a "MUST-" for TCP-AO today, and the only mandatory-to-implement MAC. AES-128 CMAC is considered to be far stronger, but may not yet have very wide implementation, it is therefore a SHOULD+.

2.3. Requirements for Future MAC Algorithms

Since this document provides cryptographic agility, it is also important to establish requirements for future MAC algorithms. The TCPM WG should restrict any future MAC algorithms for this specification to ones that can protect at least $2^{48}$ messages with a probability that a collision will occur of less than one in a billion.

[Reviewers: Are there other requirements we want to place in here?]

3. Algorithms Specified

TCP-AO refers to this document saying that the MAC mechanism employed
for a connection is listed in the TSAD entry, and is chosen from a list of MACs both named and described in this document.

TCP-AO requires two classes of cryptographic algorithms:

(1) Key Derivation Functions (KDFs) which name a pseudorandom function (PRF) and use a Master_Key and some connection-specific Input with that PRF to produce Conn_Keys, the keys suitable for authenticating and integrity checking individual TCP segments.

(2) Message Authentication Code (MAC) algorithms which take a key and a message and produce an authentication tag which can be used to verify the integrity of the message. In TCP-AO, these algorithms are always used in pairs. Each MAC algorithm MUST specify the KDF to be used with that MAC algorithm. However, a KDF MAY be used with more than one MAC algorithm.

In TCP-AO, these algorithms are always used in pairs. Each MAC algorithm MUST specify the KDF to be used with that MAC algorithm. However, a KDF MAY be used with more than one MAC algorithm.

3.1. Key Derivation Functions (KDFs)

TCP-AO’s Conn_Keys are derived using KDFs. The KDFs used in TCP-AO manual have the following interface:

PRF(Master_Key, Input, Output_Length)

where:

- PRF: the specific pseudorandom function that is the basic building block used in constructing the given KDF

- Master_Key: The Master_Key as will be stored into the associated TCP-AO TSAD entry. In TCP-AO’s manual key mode, this is a shared key that both sides enter via some user interface into their respective configurations. The Master_Key is the seed for the PRF. We assume that, in manual key mode, this is a human readable pre-shared key (PSK), thus we assume that it is of variable length, and we also assume it is in no way random.
- Input: the input data for the PRF, in conformance with [NIST-SP800-108], is a concatenation of:

\[ i \mid \text{Label} \mid 0x00 \mid \text{Context} \mid \text{Output\_Length} \]

Where

- "||": Represents a concatenation operation, between two values X || Y.

- i: A counter, a binary string that is an input to each iteration of a PRF in counter mode and (optionally) in feedback mode. This will depend on the specific size of the Output\_Length desired for a given MAC.

- Label: A binary string that clearly identifies the purpose of this KDF’s derived keying material. For TCP-AO we use the ASCII string "TCP-AO". While this may seem like overkill in this specification since TCP-AO only describes one call to the KDF, it is included in order to comply with FIPS 140 certifications.

- 0x00: Eight zero bits, or 0 represented in byte form

- Context: A binary string containing information related to the specific connection for this derived keying material. In TCP-AO, this is the Conn\_Block, as defined in [I-D.ietf-tcpm-tcp-auth-opt], Section X.

- Output\_Length: The length in bits of the key that the KDF will produce. This length must be the size required for the MAC algorithm that will use the PRF result as a seed.

When invoked, a KDF runs a certain PRF, using the Master\_Key as the seed, and Input as the message input and produces a result of Output\_Length bits. This result may then be used as a cryptographic key for any algorithm which takes an Output\_Length length key as its seed. A KDF MAY specify a maximum Output\_Length parameter.

This document defines two KDFs:
* KDF_HMAC_SHA1 based on PRF-HMAC-SHA1 [RFC2404]
* KDF_AES_128_CMAC based on AES-CMAC-PRF-128 [RFC4615]

Other KDFs may be defined in future revisions of this document, and
SHOULD follow this same format as described above.

3.1.1. The Use of KDF_HMAC_SHA1

For:

PRF(Master_Key, Input, Output_Length)

KDF_HMAC_SHA1 for TCP-AO has the following values:

- PRF: HMAC-SHA1 [RFC2404]
- Master_Key: As provided in the TSAD
- Input:
  - i: "0"
  - Label: "TCP-AO"
  - Context: Conn_Block
  - Output_Length 160
- Result: Conn_Key

The result is computed by performing HMAC-SHA1(Master_Key, Input) and
then taking the first (high order) Output_Length, 160 here, bits.
This result is the TCP-AO Conn_Key. The Conn_Key is then used as the
seed for the MAC function on each segment of the connection.

3.1.2. The Use of KDF_AES_128_CMAC

For:

PRF(Master_Key, Input, Output_Length)

KDF_AES_128_CMAC for TCP-AO has the following values:

- PRF: AES-CMAC-PRF-128 [RFC4615]
- Master_Key: As provided in the TSAD
- Input:
The result is computed by performing AES-CMAC-PRF-128(Master_Key, Input) and then taking the first (high order) Output_Length, 128, bits. This result is the TCP-AO Conn_Key. The Conn_Key is then used as the seed for the MAC function on each segment of the connection.

Since the Master_Key in the TCP-AO manual key mode is a pre-shared key (PSK) passed in an out of band mechanism between two devices, and often between two organizations, it is assumed to be of variable length. Therefore it may not have 16 octets / 128 bits, as is required as an input length to AES-128. We could mandate that implementations force administrators to input only keys of such length, and with sufficient randomness, but this places undue burden on the deployers. Instead, to make things easier on the deployers, we use the AES-CMAC-PRF-128 mechanism represented in [RFC4615], Sect 3, with a minor change: we never use the raw Master_Key (MK) alone if K := MK. Instead, we always assume MK is variable length, and we always use both the 0^128, the MK, and the MKlen arguments, even when K := MK. Therefore this KDF is always a 2 step function, as follows (borrowing the format from [RFC4615]):

```
+ Input  : MK (Variable-length Master_Key)                          +
+        : I (Input, i.e., the input data of the PRF)               +
+        : MKlen (length of MK in octets)                           +
+        : len (length of I in octets)                             +
+ Output : PRV (128-bit Pseudo-Random Variable)                   +
+                                                                   +
+ Variable: K (128-bit key for AES-CMAC)                          +
+                                                                   +
+ Step 1.     K := AES-CMAC(0^128, MK, MKlen);                     +
+ Step 2.   PRV := AES-CMAC(K, I, len);                           +
+           return PRV;                                           +
```

Figure 1: The AES-CMAC-PRF-128 Algorithm for TCP-AO
In Step 1, the 128-bit key, K, for AES-CMAC is derived by applying the AES-CMAC algorithm using the 128-bit all-zero string as the key and MK as the input message.

In Step 2, we apply the AES-CMAC algorithm again, this time using K as the key and I as the input message. The output of this algorithm returns PRV, 128 bits suitable for the seed, the Conn_Key, in the AES-CMAC operation over the segment.

### 3.2. MAC Algorithms

MACs for TCP-AO have the following interface:

MAC (Conn_Key(KDF), Message, Truncation)

where:

- MAC-algo: MAC Algorithm used
- Conn_Key: Variable; Result of KDF.
- KDF: Name of the TCP-AO KDF used
- Key_Length: Length in bits required for the Conn_Key used in this MAC
- Truncation: Length in bits to which the final MAC result is truncated before being placed into TCP-AO header

This document specifies two MAC algorithm options for generating the MAC for TCP-AO’s option header:

* HMAC-SHA-1-961 based on [RFC2404]

* AES-128-CMAC-96 based on [RFC4493]

Both provide a high level of security and efficiency. The AES-128-CMAC-96 is potentially more efficient, particularly in hardware, but HMAC-SHA-1-96 is more widely used in Internet protocols and in most cases could be supported with little or no additional code in today’s deployed software and devices.

An important aspect to note about these algorithms’ definitions for use in TCP-AO is the fact that the MAC outputs are truncated to 96 bits. AES-128-CMAC-96 produces a 128 bit MAC, and HMAC SHA-1 produces a 160 bit result. The MAC output are then truncated to 96 bits to provide a reasonable tradeoff between security and message...
size, for fitting into the TCP-AO header.

3.2.1. The Use of HMAC-SHA-1-96

By definition, HMAC [RFC2104] requires a cryptographic hash function. SHA1 will be that has function used for authenticating and providing integrity validation on TCP segments with HMAC.

For:

MAC (Conn_Key(KDF), Message, Truncation)

HMAC-SHA-1-96 for TCP-AO has the following values:

- MAC-algo: MAC Algorithm used
- Conn_Key: Variable; Result of KDF.

- KDF: KDF_HMAC_SHA1
- Key_Length: 160 bits
- Truncation: 96 bits

3.2.2. The Use of AES-128-CMAC-96

In the context of TCP-AO, when we say "AES-128-CMAC-96" we actually define a usage of AES-128 as a cipher-based MAC according to [NIST-SP800-38B].

For:

MAC (Conn_Key(KDF), Message, Truncation)

AES-128-CMAC-96 for TCP-AO has the following values:

- MAC-algo: AES-128-CMAC-96 [RFC4493]
- Conn_Key: Variable; Result of KDF.

- KDF: KDF_AES_128_CMAC
- Key_Length: 128 bits
- Truncation: 96 bits

According to [RFC4493], by default, "the length of the output of AES-128-CMAC is 128 bits. It is possible to truncate the MAC. The result of the truncation is then taken in most significant bits first order. The MAC length must be specified before the communication starts, and it must not be changed during the lifetime of the key." Therefore, we explicitly specify the employed MAC length for TCP-AO to be 96 bits.

4. UI Labels

[Note to reviewers: We may want to move this section out to a separate doc, or scrap it altogether. I already had it in here when I brought it up on the design team call, where it was decided to do it in another helper-doc in an ops wg. I didn’t have time to yank it before the -00 deadline. Let me know what you think of it. I’d like to keep it in to make it easy for a future when we have a KMP and need to define suites, but I’m more than willing to shelve it until later. Pls advise.]

Implementation experience with other security protocols employing cryptography (e.g. IPsec [4303] & TLS [5246]) in manual key mode, and with key management protocols (KMPs), has shown that there are so many choices for typical system administrators to make that it is difficult to achieve interoperability without careful pre-agreement. Accordingly, there should be a small number of named cryptographic algorithms that cover typical security policies. These algorithms may be presented in the administrative interface of the TCP-AO systems according to their labels presented here. These will be called "UI labels" ("user interface labels"). These labels are optional and do not prevent implementers from allowing individual selection of these or other security algorithms. These labels should not be considered extensions to TCP-AO, or any future KMP that may be used with TCP-AO, but instead administrative methods for describing sets of configurations.

Specifically, the transform substructure in TCP-AO (and any future KMP) must be used to give the value for each specified option regardless of whether or not UI labels are used.

Implementations that use UI labels SHOULD also provide a management interface for specifying values for individual cryptographic options. That is, it is unlikely that UI labels are a complete solution for matching the security policies of all TCP-AO users, and therefore an interface that gives a more complete set of options and technical names should be used as well.
TCP-AO implementations that use these UI labels SHOULD use the labels listed here. TCP-AO implementations SHOULD NOT use names different than those listed here for the algorithms and uses that are described, and MUST NOT use the names listed here for algorithms that do not match these values. These requirements are necessary for interoperability.

Additional labels can be defined by RFCs, or by updating this RFC. The strings used to identify UI labels are registered by IANA.

4.1. Label "Option-A"

This label matches the commonly deployed, non-deprecated (i.e. non-MD5) security used in devices today.

Protocol: TCP Authentication Option, TCP-AO [TCP-AO]
Authentication & Integrity Algorithm: HMAC-SHA-1-96
where HMAC-SHA-1-96 is defined in Section 3.2.1.

Moving to a new key by use of the [New_KeyID] field advertisement MUST be supported by both parties in this label.

This label SHOULD be the default provided in the UI. This directive should stand only until the "Option-B" label becomes widely deployed. Once devices capable of "Option-B" are widely deployed, this document should be updated to indicate "Option-B" as the default.

4.2. Label "Option-B"

This label matches the next up and coming, stronger MAC.

Protocol: TCP Authentication Option, TCP-AO [TCP-AO]
Authentication & Integrity Algorithm: AES-128-CMAC-96
where AES-128-CMAC-96 is defined in the Section 3.2.2.

Moving to a new key by use of the [New_KeyID] field advertisement MUST be supported by both parties in this label.

This label SHOULD be the second choice provided in the UI. This directive should stand only until the "Option-B" label becomes widely deployed. Once devices capable of "Option-B" are widely deployed, this document should be updated to indicate "Option-B" as the default.
5. Change History (RFC Editor: Delete before publishing)

[NOTE TO RFC EDITOR: this section for use during I-D stage only. Please remove before publishing as RFC.]

-00 - original submission
-01 - not yet done (place holder)
o adds new stuff.


[NOTE TO RFC EDITOR: this section for use during I-D stage only. Please remove before publishing as RFC.]

List of stuff that still needs work
o fix the iana registry section. Need registry entries for the KDFs and all the other values?
o

7. Security Considerations

This document inherits all of the security considerations of the TCP-AO, the AES-CMAC, and the HMAC-SHA-1 documents.

The security of cryptographic-based systems depends on both the strength of the cryptographic algorithms chosen and the strength of the keys used with those algorithms. The security also depends on the engineering of the protocol used by the system to ensure that there are no non-cryptographic ways to bypass the security of the overall system.

Care should also be taken to ensure that the selected key is unpredictable, avoiding any keys known to be weak for the algorithm in use. [RFC4086] contains helpful information on both key generation techniques and cryptographic randomness.

Note that in the composition of KDF_AES_128_CMAC, the PRF needs a 128 bit / 16 byte key as the seed. However, for convenience to the administrators/deployers, we did not want to force them to enter a 16 byte Master_Key. So we specified the sub-key routine that could handle a variable length Master_Key, one that might be less than 16 bytes. This does NOT mean that administrators are safe to use weak keys. Administrators are encouraged to follow [RFC4086] as listed above. We simply attempted to "put a fence around stupidity", in as
This document concerns itself with the selection of cryptographic algorithms for the use of TCP-AO. The algorithms identified in this document as "MUST implement" or "SHOULD implement" are not known to be broken at the current time, and cryptographic research so far leads us to believe that they will likely remain secure into the foreseeable future. Some of the algorithms may be found in the future to have properties significantly weaker than those that were believed at the time this document was produced. Expect that new revisions of this document will be issued from time to time. Be sure to search for more recent versions of this document before implementing.

8. IANA Considerations

IANA has created and will maintain a registry called, "Cryptographic Labels for TCP-AO". The registry consists of a text string and an RFC number that lists the associated transform(s). New entries can be added to the registry only after RFC publication and approval by an expert designated by the IESG.

The initial values for the new registry are:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Defined In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option-A</td>
<td>[this document as an RFC]</td>
</tr>
<tr>
<td>Option-B</td>
<td>[this document as an RFC]</td>
</tr>
</tbody>
</table>

9. Acknowledgements

Paul Hoffman, from whose [RFC 4308] I largely followed, and sometimes copied, to quickly create a very similar document for TCP-AO.

Tim Polk, whose email summarying SAAG’s guidance to TCPM on the two hash algorithms for TCP-AO is largely cut and pasted into various sections of this document.

Jeff Schiller, Donald Eastlake and the IPsec WG, whose [RFC4307] & [4305] text comprised most of the Requirements [LINK] section of this document.

(In other words, I was truly only an editor of others’ text in creating this document.)
Eric "EKR" Rescorla and Brian Weis, who brought to clarity the issues with the inputs to PRFs for the KDFs, and was of great assistance in how to structure the text, as well as the correct cryptographic decisions.

10. References

10.1. Normative References


10.2. Informative References


Author’s Address

Gregory Lebovitz
Juniper Networks, Inc.
1194 North Mathilda Ave.
Sunnyvale, CA  94089-1206
US

Phone:
Email: gregory.ietf@gmail.com