TCPM
Internet-Draft
Intended status: Standards Track
Expires: March 9, 2010

G. Lebovitz
Juniper
E. Rescorla
RTFM

September 05, 2009

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

Status of this Memo

Copyright Notice

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's current manual keying mechanism.

Table of Contents

1. Introduction 3
2. Requirements 3
2.1. Requirements Language 3
2.2. Algorithm Requirements 3
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 8
3.1.3. Tips for User Interfaces regarding KDFs 9
3.2. MAC Algorithms 10
3.2.1. The Use of HMAC-SHA-1-96 11
3.2.2. The Use of AES-128-CMAC-96 11
4. Change History (RFC Editor: Delete before publishing) 12
5. Needs Work in Next Draft (RFC Editor: Delete Before Publishing) 13
6. Security Considerations 13
7. IANA Considerations 14
8. Acknowledgements 14
9. References 15
9.1. Normative References 15
9.2. Informative References 15
Authors' Addresses 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 [RFC2104]. 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 specifies the use of those two MACs 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].

2.2. Algorithm Requirements

In this the first RFC specifying cryptography for TCP-AO, we specify two MAC algorithms. Both MUST be implemented in order for the implementation to be fully compliant with this RFC.

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

Requirement	Authentication Algorithm
MUST	HMAC-SHA-1-96 [RFC2404]
MUST	AES-128-CMAC-96 [RFC4493]

Requirement	Key Derivation Function (KDF)
MUST	KDF_HMAC_SHA1
MUST	KDF_AES_128_CMAC

NOTE EXPLAINING WHY TWO MAC ALGORITHMS WERE MANDATED:

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. AES-128 CMAC is considered to be far stronger algorithm, but may not yet have very wide implementation. It is also a "MUST" to implement, in order to drive vendors toward its use.

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 any other requirements we want/need to place in here? RFC EDITOR: Please delete this text before publishing as RFC]

3. Algorithms Specified

TCP-AO refers to this document saying that the MAC mechanism employed for a connection is listed in the TAPD 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 Traffic_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 messages sent over the wire.

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 Traffic_Keys are derived using KDFs. The KDFs used in TCP-AO's current manual keying have the following interface:

```
Derived_Key = KDF(Master_Key, Input, Output_Length)
```

where:

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

- Master_Key: The Master_Key as will be stored into the associated TCP-AO TAPD entry. In TPC-AO's manual key mode, this is a shared key that both peers enter via some user interface into their respective configurations. The Master_Key is the seed for the KDF. 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. Users SHOULD chose random strings for the Master_Key. However, we assume that some users may not.

- Input: the input data for the KDF, in conformance with [NIST-SP800-108], is a concatonation of:

(i || Label || Context || Output_Length)

Where

- "||": Represents a concatonation 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 an 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", where the last character is the capital letter "O", not to be confused with a zero. 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.

- Context : A binary string containing information related to the specific connection for this derived keying material. In TCP-AO, this is the Data_Block, as defined in [I-D.ietf-tcpm-tcp-auth-opt], Section 7.1]

- 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.

NOTE: The cited NIST document on KDFs calls for an input: (i || Label || 0x00 || Context || Output_Length). That document states that the "0x00" is an all zero octet and is "an optional data field used to indicate a separation of different variable length data fields". In our case, the "Label" is specified and fixed, thus its data field is fixed, not variable, so there is no need for the 0x00 separator. Thus, we have dropped it.

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. When doing so, note:

1. The underlying PRFs specified in this document have fixed sized

output lengths, 128 bits in the case of the AES-CMAC, and 160 bits in the case of HMAC-SHA1.

2. It is possible to generate an arbitrary number of output bits with some given PRF by operating it in a feedback or counter mode. The KDFs described in [NIST-SP800-108] incorporate this feature, hence the counter "i", which creates leading "0".
3. Each MAC needs a key of a specific length.
4. Not totally uncoincidentally, the KDFs we have chosen to use with each MAC happen to generate the right key size for use with the MAC, thus avoiding the need for the procedure in (2).
5. If one wanted to use these KDFs with a MAC requiring a longer key (e.g., HMAC-SHA-256) one would need to use the procedure: KDF_X = PRF_X(Master_Key, Input).

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 MKT
- Input:
  - i: "0"
  - Label: "TCP-AO"
  - Context: Data_Block
  - Output_Length 160
- Result: Traffic_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 Traffic_Key. The Traffic_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 MKT
- Input:
  - i: "0"
  - Label: "TCP-AO"
  - Context: Data_Block
  - Output_Length 128
- Result: Traffic_Key

Whereas the KDF_SHA1 used only one round to produce the Traffic_Key, the KDF_AES will take two steps. The reasoning follows:

The Master_Key in TCP-AO's current manual keying mechanism is a shared secret, entered by an administrator. It is passed via an out-of-band mechanism between two devices, and often between two organizations. The shared secret does not have to be 16 octets, and the length may vary. However, AES_128_CMAC requires a key of 16 octets (128 bits) in length. We could mandate that implementations force administrators to input Master_Keys of exactly 128 bit length, and with sufficient randomness, but this places undue burdon on the deployers. This specification RECOMMENDS that deployers use a randomly generated 128-bit string as the Master_Key, but acknowledges that deployers may not.

To handle variable length Master_Keys we use a similar mechanism to the AES-CMAC-PRF-128 mechanism represented in [RFC4615], Sect 3. We do a two step process.

First we use AES_128_CMAC with a fixed key as a "randomness extractor", while using the shared secret Master_Key, MK, as the message input to produce a 128 bit key K.

Second, we run AES_128_CMAC again, this time using K as the key and the normal Input I (as described above) as the message input to produce Traffic_Key, TK.

Therefore this KDF is always a 2 step function, as follows (borrowing the format from [RFC4615]):

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

Figure 1: The AES-CMAC-PRF-128 Algorithm for TCP-AO

o 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.

o In Step 2, we apply the AES-CMAC algorithm again, this time using the output of Step 1, K, as the key. The input message is now I, just as it is described at the beginning of this section. The output of this algorithm returns TK, the Traffic_Key, which is 128 bits suitable for the seed into the AES-CMAC operation over the actual TCP segment.

3.1.3. Tips for User Interfaces regarding KDFs

This section provides suggested representations for the KDFs in implementation user interfaces. Following these guidelines across common implementations will make interoperability easier and simpler for deployers.

UIs SHOULD refer to the choice of KDF_HMAC_SHA1 as simply "SHA1".

UIs SHOULD refer to the choice of KDF_AES_128_CMAC as simply "AES128".

UIs SHOULD use KDF_HMAC_SHA1 as the default selection in TCP-AO settings. KDF_HMAC_SHA1 is preferred at this time solely because it

has wide support, being present in most implementations in the marketplace. When such a time arrives as KDF_AES_128_CMAC becomes

widely deployed, this document should be updated so that it becomes the default KDF on implementations.

3.2. MAC Algorithms

MACs for TCP-AO have the following interface:

MAC (Traffic_Key(KDF), Message, Truncation)

where:

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

  - KDF: Name of the TCP-AO KDF used

- Key_Length: Length in bits required for the Traffic_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-1-96 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 (Traffic_Key(KDF), Message, Truncation)

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

- MAC-algo: MAC Algorithm used
- Traffic_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 (Traffic_Key(KDF), Message, Truncation)

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

- MAC-algo: AES-128-CMAC-96 [RFC4493]
- Traffic_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. 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.]

draft-ietf...-00 - 4th submission

Submitting draft-lebovitz...-02 as a WG document. Added EKR as co-author, and gave him credit for rewrite of sect 3.1.x .

lebovitz...-02 - 3rd submission

o cleaned up explanation in 3.1.2
o in 3.1.2, changed the key extractor mechanism back, from using an

alphanumeric string for the fixed key C to use 0^128 as the key (as it was in -00) (Polk & Ekr)
- deleted cut-and-paste error text from sect 3.1 between "label" and "context"
- changed "conn_key" to "traffic_key" throughout, to follow auth-opt-05
- changed "tsad" to "mkt" throughout, to follow auth-opt-05
- changed "conn_block" to "data_block" throughout, to follow auth-opt-05

lebovitz...-01- 2nd submission

- removed the whole section on labels (previously section 4), per WG consensus at IETF74. Added 3.1.3 to specify that implementations SHOULD make HMAC-SHA1 the default choice for the time being, and to suggest common names for the KDF's universally in UI's.
- changed KDF = PRF... to Derived_Key = KDF... (EKR)
- added the text on how to deal with future KDF to end of s3.1 (EKR)
- removed references to TCP-AO "manual key mode". Changed to TCP-AO's "current mode of manual keying". (Touch)
- removed the whole MUST- / SHOULD+ thing. Both KDF's are MUST now, per wg consensus at ietf74.

- in 3.1.2, changed the mechanism to force the K to be 128bits from using 0^128, to using a fixed 128-bit string of random characters (Dave McGrew)
- sect 3.1, in Input description, dropped "0x00". Added "NOTE" explaining why right after the output_length description.
- cleaned up all references
- copy editing

lebovitz...-00 - original submission

5. Needs Work in Next Draft (RFC Editor: Delete Before Publishing)

[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
- fix the iana registry section. Need registry entries for the KDFs

and all the other values?

- this was supposed to be named draft-ietf-tcpm-tcp-ao-crypto-00.txt, but I forgot that since we were moving from a personal submission to a wg sub, it had to go back to a -00, thus needed to be done a week earlier. Oops. Will fix as soon as the window opens for submitting -00's again.

6. 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 much as possible.

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.

7. IANA Considerations

IANA has created and will maintain a registry called, "Cryptographic Algorithms 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.

[need to finish this section]

8. Acknowledgements

Eric "EKR" Rescorla, who provided a ton of input and feedback, including a somewhat heavy re-write of section 3.1.x, earning him and author slot on the document.

Paul Hoffman, from whose [RFC4308] I sometimes copied, to quickly create a first draft here.

Tim Polk, whose email summarizing 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] & [RFC4305] text was consulted and sometimes used in the Requirements Section 2 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. EKR was also of great assistance in how to structure the text, as well as helping to guide good cryptographic decisions.

The TCPM working group, who put up with all us crypto and routing folks DoS'ing their WG for 2 years, and who provided reviews of this document.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[HMAC-ATTACK]
"On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1"", 2006, <http://eprint.iacr.org/2006/187 http://www.springerlink.com/content/00w4v62651001303>.

[I-D.ietf-tcpm-tcp-auth-opt]
Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", draft-ietf-tcpm-tcp-auth-opt-05 (work in progress), July 2009.

[I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

[NIST-SP800-108]
National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions", SP 800-108, April 2008.

[NIST-SP800-38B]
National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", SP 800-38B, May 2005.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104,

February 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4305] Eastlake, D., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305, December 2005.

[RFC4307] Schiller, J., "Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)", RFC 4307, December 2005.

[RFC4308] Hoffman, P., "Cryptographic Suites for IPsec", RFC 4308, December 2005.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, June 2006.

[RFC4615] Song, J., Poovendran, R., Lee, J., and T. Iwata, "The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)", RFC 4615, August 2006.

Authors' Addresses

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

Phone:
Email: gregory.ietf@gmail.com

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
US

Phone: 650-678-2350
Email: ekr@rtfm.com