Internet Engineering Task Force (IETF)
Request for Comments: 5926
Category: Standards Track
ISSN: 2070-1721

G. Lebovitz
Juniper
E. Rescorla
RTFM

June 2010

Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)

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 algorithms. This document specifies the algorithms and attributes that can be used in TCP-AO's current manual keying mechanism and provides the interface for future message authentication codes (MACs).

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 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5926.

Copyright Notice

Copyright (c) 2010 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 (http://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.

Table of Contents

1. Introduction 2
2. Requirements 3
   2.1. Requirements Language 3
   2.2. Algorithm Requirements 3
   2.3. Requirements for Future MAC Algorithms 3
3. Algorithms Specified 4
   3.1. Key Derivation Functions (KDFs) 4
        3.1.1. Concrete KDFs 5
               3.1.1.1. KDF_HMAC_SHA1 6
               3.1.1.2. KDF_AES_128_CMAC 7
               3.1.1.3. Tips for User Interfaces Regarding KDFs 9
   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 11
4. Security Considerations 11
5. IANA Considerations 13
6. Acknowledgements 13
7. References 14
   7.1. Normative References 14
   7.2. Informative References 14

1. Introduction

This document is a companion to [RFC5925]. Like most modern security protocols, TCP-AO allows users to choose which cryptographic algorithm(s) they want to use to meet their security needs.

TCP-AO provides cryptographic authentication and message integrity verification between two end-points. In order to accomplish this function, message authentication codes (MACs) are used, which then rely on shared keys. There are various ways to create MACs. The use of hash-based MACs (HMACs) is defined in [RFC2104]. The use of cipher-based MACs (CMACs) is defined in [NIST-SP800-38B].

This RFC defines the general requirements for MACs used in TCP-AO, both for currently specified MACs and for any future specified MACs. It specifies two MAC algorithms required in all TCP-AO implementations. It also specifies two key derivation functions (KDFs) used to create the traffic keys used by the MACs. These KDFs are also required by all TCP-AO implementations.

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].

When used in lowercase, these words convey their typical use in common language, and they are not to be interpreted as described in [RFC2119].

2.2. Algorithm Requirements

This is the initial specification of required cryptography for TCP-AO, and indicates two MAC algorithms and two KDFs. All four components MUST be implemented in order for the implementation to be fully compliant with this RFC.

The following table indicates the required MAC algorithms and KDFs for TCP-AO:

Requirement	Authentication Algorithm
MUST	HMAC-SHA-1-96 [RFC2104][FIPS-180-3]
MUST	AES-128-CMAC-96 [NIST-SP800-38B][FIPS197]

Requirement	Key Derivation Function (KDF)
MUST	KDF_HMAC_SHA1
MUST	KDF_AES_128_CMAC

For an explanation of why two MAC algorithms were mandated, see the Section 4.

2.3. Requirements for Future MAC Algorithms

TCP-AO is intended to support cryptographic agility. As a result, this document includes recommendations in various places for future MAC and KDF algorithms when used for TCP-AO. For future MAC algorithms specifically, they SHOULD protect at least 2**48 messages with a collision probability of less than one in 10**9.

3. Algorithms Specified

TCP-AO requires two classes of cryptographic algorithms used on a particular connection, and refers to this document to define them both:

(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, as described in TCP-AO.

(2) Message Authentication Code (MAC) algorithms, which take a key and a message and produce an authentication tag that can be used to verify the integrity and authenticity of those messages.

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:

```
Traffic_Key = KDF_alg(Master_Key, Context, Output_Length)
```

where:

- KDF_alg: the specific pseudorandom function (PRF) that is the basic building block used in constructing the given Traffic_Key.

- Master_Key: In TCP-AO's manual key mode, this is a key shared by both peers, entered via some interface to their respective configurations. The Master_Key is used as the seed for the KDF. We assume that this is a human-readable pre-shared key (PSK); thus, we assume it is of variable length. Master_Keys SHOULD be random, but might not be (e.g., badly chosen by the user). For interoperability, the management interface by which the PSK is configured MUST accept ASCII strings, and SHOULD also allow for the configuration of any arbitrary binary string in hexadecimal form. Other configuration methods MAY be supported.

- Context: A binary string containing information related to the specific connection for this derived keying material, as defined in [RFC5925], Section 5.2.

- 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 generates a string of length Output_Length bits based on the Master_Key and context value. This result may then be used as a cryptographic key for any algorithm that takes an Output_Length length key. A KDF MAY specify a maximum Output_Length parameter.

3.1.1. Concrete KDFs

This document defines two KDF algorithms, each paired with a corresponding PRF algorithm as explained below:

* KDF_HMAC_SHA1 based on PRF-HMAC-SHA1 [RFC2104][FIPS-180-3]

* KDF_AES_128_CMAC based on AES-CMAC-PRF-128 [NIST-SP800-38B][FIPS197]

Both of these KDFs are based on the iteration-mode KDFs specified in [NIST-SP800-108]. This means that they use an underlying pseudorandom function (PRF) with a fixed-length output, 128 bits in the case of the AES-CMAC, and 160 bits in the case of HMAC-SHA1. The KDF generates an arbitrary number of output bits by operating the PRF in a "counter mode", where each invocation of the PRF uses a different input block differentiated by a block counter.

Each input block is constructed as follows:

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

Where

- "||": For any X || Y, "||" represents a concatenation operation of the binary strings X and Y.

- i: A counter, a binary string that is an input to each iteration of the PRF in counter mode. The counter "i" is represented in a single octet. The number of iterations will depend on the specific size of the Output_Length desired for a given MAC. "i" always starts = 1.

- 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: The context argument provided to the KDF interface, as described above in Section 3.1 .

- Output_Length: The length, in bits, of the key that the KDF will produce. The Output_length is represented within two octets. This length must be the size required for the MAC algorithm that will use the PRF result as a seed.

The output of multiple PRF invocations is simply concatenated. For the Traffic_Key, values of multiple PRF invocations are concatenated and truncated as needed to create a Traffic_Key of the desired length. For instance, if one were using KDF_HMAC_SHA1, which uses a 160-bit internal PRF to generate 320 bits of data, one would execute the PRF twice, once with i=1 and once with i=2. The result would be the entire output of the first invocation concatenated with the second invocation. For example,

```
Traffic_Key =
        KDF_alg(Master_Key, 1 || Label || Context || Output_length) ||
        KDF_alg(Master_Key, 2 || Label || Context || Output_length)
```

If the number of bits required is not an exact multiple of the output size of the PRF, then the output of the final invocation of the PRF is truncated as necessary.

3.1.1.1. KDF_HMAC_SHA1

For KDF_HMAC_SHA1:

- PRF for KDF_alg: HMAC-SHA1 [RFC2104][FIPS-180-3].

- Use: HMAC-SHA1(Key, Input).

- Key: Master_Key, configured by user, and passed to the KDF.

- Input: (i || Label || Context || Output_Length)

- Output_Length: 160 bits.

- Result: Traffic_Key, used in the MAC function by TCP-AO.

3.1.1.2. KDF_AES_128_CMAC

For KDF_AES_128_CMAC:

- PRF for KDF_alg: AES-CMAC-PRF-128 [NIST-SP800-38B][FIPS197].

- Use: AES-CMAC(Key, Input).

- Key: Master_Key (see usage below)

- Input: (i || Label || Context || Output_Length)

- Output_Length: 128 bits.

- Result: Traffic_Key, used in the MAC function by TCP-AO

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 exactly 16 octets (128 bits) in length. We could mandate that implementations force administrators to input Master_Keys of exactly 128-bit length when using AES_128_CMAC, and with sufficient randomness, but this places undue burden on the implementors and 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 the same mechanism as described in [RFC4615], Section 3. First, we use AES_128_CMAC with a fixed key of all zeros as a "randomness extractor", while using the shared secret Master_Key, MK, as the message input, to produce a 128-bit key Derived_Master_Key (K). Second, we use the result as a key, and run AES-128_CMAC again, this time using the result K as the Key, and the true input block as the Input to yield the Traffic_Key (TK) used in the MAC over the message. The description follows:

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+                        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 M in octets)                              +
+ Output : TK (Traffic_Key, 128-bit Pseudo-Random Variable)         +
+                                                                    +
+-------------------------------------------------------------------+
+ Variable: K (128-bit key for AES-CMAC)                             +
+                                                                    +
+ Step 1.   If MKlen is equal to 16                                 +
+ Step 1a.  then                                                    +
+               K := MK;                                             +
+ Step 1b.  else                                                    +
+               K := AES-CMAC(0^128, MK, MKlen);                     +
+ Step 2.   TK := AES-CMAC(K, I, len);                               +
+           return TK;                                               +
+                                                                    +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 1

In step 1, the 128-bit key, K, for AES-CMAC is derived as follows:

o If the Master_Key, MK, provided by the administrator is exactly 128 bits, then we use it as is.

o If it is longer or shorter than 128 bits, then we derive the key K by applying the AES-CMAC algorithm using the 128-bit all-zero string as the key and MK as the input message. This step is described in 1b.

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 TK, the Traffic_Key, which is 128 bits and is suitable for use in the MAC function on each TCP segment of the connection.

3.1.1.3. Tips for User Interfaces Regarding KDFs

This section provides suggested representations for the KDFs in implementation user interfaces (UIs). 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".

The initial IANA registry values reflect these two entries.

UIs SHOULD use KDF_HMAC_SHA1 as the default selection in TCP-AO settings. KDF_HMAC_SHA1 is preferred at this time because it has wide support, being present in most implementations in the marketplace.

3.2. MAC Algorithms

Each MAC_alg defined for TCP-AO has three fixed elements as part of its definition:

- KDF_Alg: Name of the TCP-AO KDF algorithm used to generate the Traffic_Key.

- Key_Length: Length, in bits, required for the Traffic_Key used in this MAC.

- MAC_Length: The final length of the bits used in the TCP-AO MAC field. This value may be a truncation of the MAC function’s original output length.

MACs computed for TCP-AO have the following interface:

```
MAC = MAC_alg(Traffic_Key, Message)
```

where:

- MAC_alg: MAC Algorithm used.

- Traffic_Key: Variable; the result of KDF.

- Message The message to be authenticated, as specified in [RFC5925], Section 5.1.

This document specifies two MAC algorithm options for generating the MAC as used by TCP-AO:

* HMAC-SHA-1-96 based on [RFC2104] and [FIPS-180-3].

* AES-128-CMAC-96 based on [NIST-SP800-38B][FIPS197]

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 is then truncated to 96 bits to provide a reasonable trade-off between security and message size, for fitting into the TCP-AO option field.

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

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

The three fixed elements for HMAC-SHA-1-96 are:

- KDF_Alg: KDF_HMAC_SHA1.

- Key_Length: 160 bits.

- MAC_Length: 96 bits.

For:

```
MAC = MAC_alg (Traffic_Key, Message)
```

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

- MAC_alg: HMAC-SHA1.

- Traffic_Key: Variable; the result of the KDF.

- Message: The message to be authenticated, as specified in [RFC5925], Section 5.1.

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].

The three fixed elements for AES-128-CMAC-96 are:

- KDF_Alg: KDF_AES_128_CMAC.

- Key_Length: 128 bits.

- MAC_Length: 96 bits.

For:

```
MAC = MAC_alg (Traffic_Key, Message)
```

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

- MAC_alg: AES-128-CMAC-96. [NIST-SP800-38B]

- Traffic_Key: Variable; the result of the KDF.

- Message: The message to be authenticated, as specified in [RFC5925], Section 5.1.

4. Security Considerations

This document inherits all of the security considerations of the TCP-AO [RFC5925], the AES-CMAC [RFC4493], and the HMAC-SHA-1 [RFC2104] documents.

The security of cryptography-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 it is safe for administrators to use weak keys. Administrators are encouraged to follow [RFC4086] as listed above. We simply attempted to "put a fence around foolishness", 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" 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.

NOTE EXPLAINING WHY TWO MAC ALGORITHMS WERE MANDATED:

Two MAC algorithms and two corresponding KDFs are mandated as a result of discussion in the TCPM WG, and in consultation with the Security Area Directors. SHA-1 was selected because it is widely deployed and currently has sufficient strength and reasonable computational cost, so it is a "MUST" for TCP-AO today. The security strength of SHA-1 HMACs should be sufficient for the foreseeable future, especially given that the tags are truncated to 96 bits.

Recently exposed vulnerabilities in other MACs (e.g., MD5 or HMAC MD5) aren’t practical on HMAC-SHA-1, but these types of analyses are mounting and could potentially pose a concern for HMAC forgery if they were significantly improved, over time. The security issues driving the migration from SHA-1 to SHA-256 for digital signatures [HMAC-ATTACK] do not immediately render SHA-1 weak for this application of SHA-1 in HMAC mode.

AES-128 CMAC is considered to be a stronger algorithm than SHA-1, but may not yet have very wide implementation. AES-128 CMAC is also a "MUST" to implement, in order to drive vendors toward its use, and to allow for another MAC option, if SHA-1 were to be compromised.

5. IANA Considerations

IANA has created the following registry (http://www.iana.org).

Registry Name: Cryptographic Algorithms for TCP-AO Registration
Procedure: RFC Publication after Expert Review

Initial contents of this registry are:

Algorithm	Reference
SHA1	[RFC5926]
AES128	[RFC5926]

6. 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 an author slot on the document.

Paul Hoffman, from whose [RFC4308] I sometimes copied, to quickly create a first version 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] & [RFC4835] text was consulted and sometimes used in the Requirements Section 2 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.

7. References

7.1. Normative References

[FIPS-180-3] FIPS Publication 180-3, "Secured Hash Standard", FIPS 180-3, October 2008.

[FIPS197] FIPS Publications 197, "Advanced Encryption Standard (AES)", FIPS 197, November 2001.

[NIST-SP800-108] National Institute of Standards and Technology, "Recommendation for Key Derivation Using Pseudorandom Functions, NIST SP800-108", SP 800- 108, October 2009.

[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.

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

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

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

7.2. Informative References

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

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

[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.

[RFC4835] Manral, V., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4835, April 2007.

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