TCPM Working Group
Internet-Draft
Intended status: Standards Track
Expires: 30 November 2026

R. Bonica
T. Li
HPE

29 May 2026

Additional Cryptographic Algorithms For Use With TCP-AO
draft-ietf-tcpm-tcp-ao-algs-03

Abstract

RFC5926 creates a list of cryptographic algorithms that can be used with TCP-AO. This document expands that list, adding two Message Authentication Code (MAC) algorithms, HMAC-SHA256-128 and KMAC256-128. For each MAC algorithm, a corresponding Key Derivation Function (KDF) is also added.

The MAC algorithms described by this document produce 128-bit (i.e., 16-byte) MACs. When 16-byte MACs are encoded in TCP-AO, the TCP-AO consumes 20 bytes. This does not challenge TCP's 40-byte option size limitation.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 30 November 2026.

Table of Contents

1. Introduction

[RFC5926] creates a list of cryptographic algorithms that can be used with TCP-AO [RFC5925]. This document expands that list, adding two Message Authentication Code (MAC) algorithms, HMAC-SHA256-128 and KMAC256-128. For each MAC algorithm, a corresponding Key Derivation Function (KDF) is also added.

The MAC algorithms described by this document produce 128-bit (i.e., 16-byte) MACs. When 16-byte MACs are encoded in TCP-AO, the TCP-AO consumes 20 bytes. This does not challenge TCP's [RFC9293] 40-byte option size limitation.

2. Requirements Language

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Algorithm Classes

[RFC5925] requires the following cryptographic algorithm classes:

* Key Derivation Functions (KDFs)

* MAC Algorithms

Section 3.1 of this document addresses KDFs while Section 3.2 addresses MAC algorithms.

3.1. Key Derivation Functions (KDFs)

A KDF converts Input Keying Material (IKM) into cryptographically secure Output Keying Material (OKM). In the case of TCP-AO, a KDF converts an administratively assigned Master_Key into a Traffic_Key.

KDFs have the following interface:

* Traffic_Key = KDF_alg(Master_Key, Context, Output_Length)

where:

* KDF_alg is the KDF algorithm being used.

* Master_Key is a variable length pre-shared key (PSK).

* Context is binary string containing information related to the TCP connection, as defined in [RFC5925], Section 5.2.

* Output_Length is the desired length of the Traffic_Key. In this document, the Output_Length is always equal to 256 bits.

This document defines two KDFs:

* HKDF-SHA256

* KMAC256-KDF

Section 3.1.1 of this document describes HKDF-SHA256 while Section 3.1.2 describes KMAC256-KDF.

3.1.1. HKDF-SHA256

HKDF-SHA256 is as described in [RFC5869]. HKDF-SHA256 executes in the following stages:

* Extract

* Expand

The interface to the Extract stage is:

* PRK = HKDF-Extract(salt, IKM)

where:

* PRK is a Pseudo-random key, to be used in the Expand stage.

* salt is an all-zero byte string whose length equals 32 bytes.

* IKM is the Master_Key argument provided to the KDF interface.

According to [RFC5869], the goal of the extract stage is to concentrate the possibly dispersed entropy of the input keying material into a short, but cryptographically strong pseudorandom key. Implementations MUST execute the extract stage.

The interface to the Expand stage is:

* OKM = HKDF-Expand(PRK, info, L)

where:

* OKM is the Traffic_Key.

* PRK is the value produced by the Extract stage.

* info is the Context argument provided to the KDF interface.

* L is the Output_length argument provided to the KDF interface divided by 8. (The Output_length argument provided to the KDF interface is measured in bits, while L is measured in bytes.)

The expand stage expands the pseudorandom key to the desired length. The output key length depend on the specific cryptographic algorithms for which the keys are needed. Implementations MUST execute the expand stage.

3.1.2. KMAC256-KDF

KMAC256-KDF is as described in [DOI.10.6028_NIST.SP.800-56Cr2]. So, the interface to KMAC256-KDF as described in [DOI.10.6028_NIST.SP.800-56Cr2]:

* OKM = KMAC256(Z, salt, x, H_outputBits, S)

where:

* Z is is the Master_Key argument provided to the KDF interface.

* salt is an all-zero byte string whose length equals 132 bytes.

* x is the Context argument provided to the KDF interface.

* H_outputBits is is the Output_Length argument provided to the KDF interface.

* S is the byte string 01001011 || 01000100 || 01000110, which represents the sequence of characters "K", "D," and "F" in 8-bit ASCII.

3.2. MAC Algorithms

Each MAC algorithm defined for TCP-AO has the following fixed elements as part of its definition:

* KDF_Alg is the name of the KDF algorithm used to generate the Traffic_Key.

* Key_Length is the length of the Traffic_Key used in this MAC, measured in bits. In this document, the Key_Length is always 256 bits.

* MAC_Length is the desired length of the MAC to be produced by the algorithm. In this document, the MAC_Length is always 128 bits.

MACs computed for TCP-AO have the following interface:

* MAC = MAC_alg(Traffic_Key, Message)

where:

* MAC is the value to be encoded in TCP-AO.

* MAC_alg is MAC Algorithm used.

* Traffic_Key is the result of KDF.

* Message is the message to be authenticated, as specified in [RFC5925], Section 5.1.

3.2.1. The Use of HMAC-SHA256-128

The three fixed elements for HMAC-SHA256-128 are:

* KDF_Alg: HKDF-SHA256.

* Key_Length: 256 bits.

* MAC_Length: 128 bits.

For:

* MAC = MAC_alg (Traffic_Key, Message)

HMAC-SHA256-128 for TCP-AO has the following values:

* MAC is the value to be encoded in TCP-AO.

* MAC_alg is HMAC-SHA256.

* Traffic_Key is the result of the KDF.

* Message is the message to be authenticated, as specified in [RFC5925], Section 5.1.

3.2.2. The Use of KMAC256-128

The three fixed elements for KMAC256-128 are:

* KDF_Alg: KMAC256-KDF

* Key_Length: 256 bits.

* MAC_Length: 128 bits.

For:

* MAC = MAC_alg (Traffic_Key, Message)

KMAC256-128 for TCP-AO has the following values:

* MAC is the value to be encoded in TCP-AO.

* MAC_alg is KMAC256.

* Traffic_Key is the result of the KDF.

* Message is the message to be authenticated, as specified in [RFC5925], Section 5.1.

4. Security Considerations

This document inherits all of the security considerations of [RFC5869], [RFC5925], [RFC8702], and [RFC9688].

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.

Master_Keys SHOULD have at least 256 bits of entropy. This document RECOMMENDS that operators use Master_Keys generated by a cryptographic random number generator, or similar. However, it is understood that that may not do so.

TCP-AO Master Key Tuples MUST be rotated at a rate commensurate with the strength of the cryptographic algorithms.

5. IANA Considerations

IANA is requested to add the following entries to the "Cryptographic Algorithms for TCP-AO Registration" (https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml#tcp-parameters-3).

Algorithm	Reference
HMAC-SHA256-128	This Document
KMAC256-128	This Document

Table 1: IANA Actions

6. Acknowledgements

Thanks to Eric Biggers, Lars Eggert, Gorry Fairhurst, C.M. Heard, Russ Housley, John Mattsson, Yoshifumi Nishida, Joe Touch, Michael Tuxen, and Magnus Westerlund for their review and comments.

7. Normative References

[DOI.10.6028_NIST.SP.800-56Cr2]
Barker, E., Chen, L., and R. Davis, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes", National Institute of Standards and Technology, DOI 10.6028/nist.sp.800-56cr2, August 2020, <https://doi.org/10.6028/nist.sp.800-56cr2>.

[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/rfc/rfc2119>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/rfc/rfc5925>.

[RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, DOI 10.17487/RFC5926, June 2010, <https://www.rfc-editor.org/rfc/rfc5926>.

[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/rfc/rfc8174>.

[RFC8702] Kampanakis, P. and Q. Dang, "Use of the SHAKE One-Way Hash Functions in the Cryptographic Message Syntax (CMS)", RFC 8702, DOI 10.17487/RFC8702, January 2020, <https://www.rfc-editor.org/rfc/rfc8702>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/rfc/rfc9293>.

[RFC9688] Housley, R., "Use of the SHA3 One-Way Hash Functions in the Cryptographic Message Syntax (CMS)", RFC 9688, DOI 10.17487/RFC9688, November 2024, <https://www.rfc-editor.org/rfc/rfc9688>.

Authors' Addresses

Ron Bonica
HPE
United States of America
Email: ronald.bonica@hpe.com

Tony Li
HPE
United States of America
Email: tony.li@tony.li