TCPM
Internet Draft
Intended status: Informational
Expires: December 2020

J. Touch
Independent consultant
J. Kuusisaari
Infinera

June 11, 2020

TCP-AO Test Vectors

draft-touch-tcpm-ao-test-vectors-00.txt

Status of this Memo

Copyright Notice

Abstract

This document provides test vectors to validate implementations of the two mandatory authentication algorithms specified for the TCP Authentication Option over both IPv4 and IPv6. This includes validation of the key derivation function (KDF) based on a set of test connection parameters as well as validation of the message authentication code (MAC). Vectors are provided for both currently required pairs of KDF and MAC algorithms: one based on SHA-1 and the other on AES-128. The vectors also validate both whole TCP segments as well as segments whose options are excluded for NAT traversal.

Table of Contents

1. Introduction

This document provides test vectors to validate the correct implementation of the TCP Authentication Option (TCP-AO) [RFC5925]. It includes the specification of all endpoint parameters to generate the variety of TCP segments covered by different keys and MAC coverage, i.e., both the default case and the variant where TCP options are ignored. It also includes both default key derivation functions (KDFs) and MAC generation algorithms [RFC5926].

The experimental extension to support NAT traversal is not included in the provided test vectors [RFC6978].

This document provides test vectors from an implementation.

2. Conventions used in this document

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. Background

(TBD)

4. Input Test Vectors

4.1. TCP Connection Parameters

The following parameters are used throughout this suite of test vectors. The terms 'active' and 'passive' are used as defined for TCP [RFC793].

4.1.1. TCP-AO parameters

The following values are used for all exchanges. This suite does not test key switchover. The KeyIDs are as indicated for TCP-AO [RFC5925]. The Master Key is used to derive the traffic keys [RFC5926].

Active (client) side KeyID: 61 (3D)

Passive (server) side KeyID: 84 (54)

Master_key: "testvector" (length = 10 bytes)

4.1.2. Active (client) side parameters

The following endpoint parameters are used on the active side of the TCP connection, i.e., the side that initiates the TCP SYN.

For IPv4: 10.11.12.13

For IPv6: (TBD)

TCP port: 57969 (E271)

4.1.3. Passive (server) side parameters

The following endpoint parameters are used for the passive side of the TCP connection, i.e., the side that responds with a TCP SYN-ACK.

For IPv4: 172.27.28.29

For IPv6: (TBD)

TCP port = 179 (BGP)

4.1.4. Other IP fields and options

No IP options are used in these test vectors.

All IPv4 packets use the following other parameters [RFC791]: DSCP = 111000 (CS7) as is typical for BGP, ECN = 00, set DF, clear MF, and TTL of 255.

All IPv6 use the following other parameters [RFC8200]: (TBD).

4.1.5. Other TCP fields and options

The SYN and SYN-ACK segments include MSS [RFC793], NOP, WindowScale [RFC7323], SACK Permitted [RFC2018], TimeStamp [RFC7323], and TCP-AO [RFC5925], in that order.

All other example segments include NOP, NOP, TimeStamp, and TCP-AO, in that order.

All segment URG pointers are zero [RFC793]. All segments with data set the PSH flag [RFC793].

5. IPv4 SHA-1 Output Test Vectors

SHA-1 is computed as specified for TCP-AO [RFC5926].

5.1. SHA-1 MAC (default - covers TCP options)

5.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

```
01 23 a5 93 b9 db 70 62 9b be 2c a6 77 cd fd ea
6f e0 ac ad
```

IPv4/TCP:

```
45 e0 00 4c dd 0f 40 00 ff 06 bf 6b 0a 0b 0c 0d
ac 1b 1c 1d e9 d7 00 b3 fb fb ab 5a 00 00 00 00
e0 02 ff ff ca c4 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 00 15 5a b7 00 00 00 00 1d 10 3d 54
9b 6e 83 27 8b 87 96 fc c6 a9 10 04
```

MAC:

```
9b 6e 83 27 8b 87 96 fc c6 a9 10 04
```

5.1.2. Receive (server) SYN-ACK (covers options)

Receive_SYN_traffic_key:

```
b8 25 18 91 5c 07 36 3e f3 5d 1e dc 20 01 55 65
ab 71 9f cc
```

IPv4/TCP:

```
45 e0 00 4c 65 06 40 00 ff 06 37 75 ac 1b 1c 1d
0a 0b 0c 0d 00 b3 e9 d7 11 c1 42 61 fb fb ab 5b
e0 12 ff ff 37 76 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 84 a5 0b eb 00 15 5a b7 1d 10 54 3d
b9 b0 2d 41 f5 54 95 52 af 3d 8e 92
```

MAC:

```
b9 b0 2d 41 f5 54 95 52 af 3d 8e 92
```

5.1.3. Send (client) non-SYN (covers options)

Send_other_traffic_key:

```
18 a6 58 a6 6b 16 12 33 f2 51 04 6d ee 5c fe 4b
72 fd fc b7
```

IPv4/TCP:

```
45 e0 00 87 36 a1 40 00 ff 06 65 9f 0a 0b 0c 0d
ac 1b 1c 1d e9 d7 00 b3 fb fb ab 5b 11 c1 42 62
c0 18 01 04 a1 62 00 00 01 01 08 0a 00 15 5a c1
84 a5 0b eb 1d 10 3d 54 5e a5 b4 22 8b 89 81 39
9a be 66 8f ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da bf 00 b4 0a 0b 0c 0d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da bf 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
5e a5 b4 22 8b 89 81 39 9a be 66 8f
```

5.1.4. Receive (server) non-SYN (covers options)

Receive_other_traffic_key:

```
b8 25 18 91 5c 07 36 3e f3 5d 1e dc 20 01 55 65
ab 71 9f cc
```

IPv4/TCP:

```
45 e0 00 87 1f a9 40 00 ff 06 7c 97 ac 1b 1c 1d
0a 0b 0c 0d 00 b3 e9 d7 11 c1 42 62 fb fb ab 9e
c0 18 01 00 40 0c 00 00 01 01 08 0a 84 a5 0b f5
00 15 5a c1 1d 10 54 3d cb cf ef 1f d9 c1 ca 69
fd ec 5a d3 ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da c0 00 b4 ac 1b 1c 1d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da c0 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
cb cf ef 1f d9 c1 ca 69 fd ec 5a d3
```

5.2. SHA-1 MAC (omits TCP options)

5.2.1. Send (client) SYN (omits options)

Send_SYN_traffic_key:

```
91 a5 6b bc 7e 63 8f 5c 7c 3f 11 5d b6 6c 31 56
5d 1b 35 93
```

IPv4/TCP:

```
45 e0 00 4c 53 99 40 00 ff 06 48 e2 0a 0b 0c 0d
ac 1b 1c 1d ff 12 00 b3 cb 0e fb ee 00 00 00 00
e0 02 ff ff 54 1f 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 00 02 4c ce 00 00 00 00 1d 10 3d 54
c2 e0 94 29 95 2a 2c ee 66 62 e8 01
```

MAC:

```
c2 e0 94 29 95 2a 2c ee 66 62 e8 01
```

5.2.2. Receive (server) SYN-ACK (omits options)

Receive_SYN_traffic_key:

```
0c 09 bb 47 01 46 3f df 8e 6d 44 04 33 ed 95 3c
8e d7 05 2e
```

IPv4/TCP:

```
45 e0 00 4c 32 84 40 00 ff 06 69 f7 ac 1b 1c 1d
0a 0b 0c 0d 00 b3 ff 12 ac d5 b5 e1 cb 0e fb ef
e0 12 ff ff 38 8e 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 57 67 72 f3 00 02 4c ce 1d 10 54 3d
46 b0 af 02 e7 e3 71 eb 7d 59 72 30
```

MAC:

```
46 b0 af 02 e7 e3 71 eb 7d 59 72 30
```

5.2.3. Send (client) non-SYN (omits options)

Send_other_traffic_key:

```
10 c8 4a f5 61 47 73 0f 4c 3f 6a 51 23 99 8b 33
8e d7 a4 21
```

IPv4/TCP:

```
45 e0 00 87 a8 f5 40 00 ff 06 f3 4a 0a 0b 0c 0d
ac 1b 1c 1d ff 12 00 b3 cb 0e fb ef ac d5 b5 e2
c0 18 01 04 6c 45 00 00 01 01 08 0a 00 02 4c ce
57 67 72 f3 1d 10 3d 54 d1 1e 8f b9 44 59 15 b6
de 08 4d ca ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da bf 00 b4 0a 0b 0c 0d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da bf 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
d1 1e 8f b9 44 59 15 b6 de 08 4d ca
```

5.2.4. Receive (server) non-SYN (omits options)

Receive_other_traffic_key:

```
0c 09 bb 47 01 46 3f df 8e 6d 44 04 33 ed 95 3c
8e d7 05 2e
```

IPv4/TCP:

```
45 e0 00 87 54 37 40 00 ff 06 48 09 ac 1b 1c 1d
0a 0b 0c 0d 00 b3 ff 12 ac d5 b5 e2 cb 0e fc 32
c0 18 01 00 46 b6 00 00 01 01 08 0a 57 67 72 f3
00 02 4c ce 1d 10 54 3d 8a 3c b4 d4 59 a0 e3 75
08 03 bd d5 ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da c0 00 b4 ac 1b 1c 1d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da c0 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
8a 3c b4 d4 59 a0 e3 75 08 03 bd d5
```

6. IPv4 AES-128 Output Test Vectors

AES-128 is computed as required by TCP-AO [RFC5926].

6.1. AES MAC (default - covers TCP options)

6.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

```
74 c6 31 f0 eb c7 73 a6 08 2b 4e e0 e0 32 3a 33
```

IP/TCP:

```
45 e0 00 4c 7b 9f 40 00 ff 06 20 dc 0a 0b 0c 0d
ac 1b 1c 1d c4 fa 00 b3 78 7a 1d df 00 00 00 00
e0 02 ff ff 5a 0f 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 00 01 7e d0 00 00 00 00 1d 10 3d 54
d0 95 02 c7 53 45 67 73 3b 93 d0 a9
```

MAC:

```
d0 95 02 c7 53 45 67 73 3b 93 d0 a9
```

6.1.2. Receive (server) SYN-ACK (covers options)

Receive_SYN_traffic_key:

```
d6 c3 a1 bc 5a ce 71 52 4c 95 99 7e dd 82 07 6a
```

IPv4/TCP:

```
45 e0 00 4c 4b ad 40 00 ff 06 50 ce ac 1b 1c 1d
0a 0b 0c 0d 00 b3 c4 fa fa dd 6d e9 78 7a 1d e0
e0 12 ff ff f3 f2 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 93 f4 e9 e8 00 01 7e d0 1d 10 54 3d
fa 41 0a b2 1d 37 d4 2e 73 72 99 03
```

MAC:

```
fa 41 0a b2 1d 37 d4 2e 73 72 99 03
```

6.1.3. Send (client) non-SYN (covers options)

Send_other_traffic_key:

```
65 47 27 3f 54 f5 31 00 1a ac bc b5 4d 81 ca 63
```

IPv4/TCP:

```
45 e0 00 87 fb 4f 40 00 ff 06 a0 f0 0a 0b 0c 0d
ac 1b 1c 1d c4 fa 00 b3 78 7a 1d e0 fa dd 6d ea
c0 18 01 04 95 05 00 00 01 01 08 0a 00 01 7e d0
93 f4 e9 e8 1d 10 3d 54 79 5d 2d 3a 10 38 da 87
fd 8a ae 40 ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da bf 00 b4 0a 0b 0c 0d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da bf 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
79 5d 2d 3a 10 38 da 87 fd 8a ae 40
```

6.1.4. Receive (server) non-SYN (covers options)

Receive_other_traffic_key:

```
d6 c3 a1 bc 5a ce 71 52 4c 95 99 7e dd 82 07 6a
```

IPv4/TCP:

```
45 e0 00 87 b9 14 40 00 ff 06 e3 2b ac 1b 1c 1d
0a 0b 0c 0d 00 b3 c4 fa fa dd 6d ea 78 7a 1e 23
c0 18 01 00 e7 db 00 00 01 01 08 0a 93 f4 e9 e8
00 01 7e d0 1d 10 54 3d 23 60 70 a2 91 10 e2 a9
6e be a9 87 ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da c0 00 b4 ac 1b 1c 1d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da c0 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
23 60 70 a2 91 10 e2 a9 6e be a9 87
```

6.2. AES MAC (omits TCP options)

6.2.1. Send (client) SYN (omits options)

Send_SYN_traffic_key:

```
26 aa 6e 0c 02 c0 8e 6d 7c eb b9 46 1d d8 ec 63
```

IPv4/TCP:

```
45 e0 00 4c f2 2e 40 00 ff 06 aa 4c 0a 0b 0c 0d
ac 1b 1c 1d da 1c 00 b3 38 9b ed 71 00 00 00 00
e0 02 ff ff 70 bf 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a 00 01 85 e1 00 00 00 00 1d 10 3d 54
5d 28 b1 1c aa 52 67 00 fa 0b be 17
```

MAC:

```
5d 28 b1 1c aa 52 67 00 fa 0b be 17
```

6.2.2. Receive (server) SYN-ACK (omits options)

Receive_SYN_traffic_key:

```
83 c6 6b 5f c1 06 4b 34 87 4a 63 45 4b 6f dd 67
```

IPv4/TCP:

```
45 e0 00 4c 6c c0 40 00 ff 06 2f bb ac 1b 1c 1d
0a 0b 0c 0d 00 b3 da 1c d3 84 4a 6f 38 9b ed 72
e0 12 ff ff e4 45 00 00 02 04 05 b4 01 03 03 08
04 02 08 0a ce 45 98 38 00 01 85 e1 1d 10 54 3d
ea a5 c4 87 c2 81 90 92 45 9e 80 e8
```

MAC:

```
ea a5 c4 87 c2 81 90 92 45 9e 80 e8
```

6.2.3. Send (client) non-SYN (omits options)

Send_other_traffic_key:

```
b1 a5 77 89 47 ef 50 2e 8f 9f 4b 7f c5 1f e6 d2
```

IPv4/TCP:

```
45 e0 00 87 ee 91 40 00 ff 06 ad ae 0a 0b 0c 0d
ac 1b 1c 1d da 1c 00 b3 38 9b ed 72 d3 84 4a 70
c0 18 01 04 88 51 00 00 01 01 08 0a 00 01 85 e1
ce 45 98 38 1d 10 3d 54 c8 0d 50 73 90 1d 29 3e
e6 0c 48 39 ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da bf 00 b4 0a 0b 0c 0d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da bf 02 08 40
06 00 64 00 01 01 00
```

MAC:

```
c8 0d 50 73 90 1d 29 3e e6 0c 48 39
```

6.2.4. Receive (server) non-SYN (omits options)

Receive_other_traffic_key:

```
83 c6 6b 5f c1 06 4b 34 87 4a 63 45 4b 6f dd 67
```

IPv4/TCP:

```
45 e0 00 87 6a 21 40 00 ff 06 32 1f ac 1b 1c 1d
0a 0b 0c 0d 00 b3 da 1c d3 84 4a 70 38 9b ed 72
c0 18 01 00 04 49 00 00 01 01 08 0a ce 45 98 38
00 01 85 e1 1d 10 54 3d f0 22 1b 32 97 a4 6e e0
a7 80 00 ca ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff 00 43 01 04 da c0 00 b4 ac 1b 1c 1d
26 02 06 01 04 00 01 00 01 02 02 80 00 02 02 02
00 02 02 42 00 02 06 41 04 00 00 da c0 02 08 40
06 00 64 00 01 01 00
```

MAC:

f0 22 1b 32 97 a4 6e e0 a7 80 00 ca

7. IPv6 SHA-1 Output Test Vectors

(TBD)

8. IPv6 AES-128 Output Test Vectors

(TBD)

9. Observed Implementation Errors

The following is a partial list of implementation errors that this set of test vectors is intended to validate.

9.1. Algorithm issues

- Underlying implementation of HMAC SHA1 or AES128 CMAC does not pass their corresponding test vectors [RFC2202] [RFC4493]

- SNE algorithm does not consider corner cases (the pseudocode in [RFC5925] was not intended as complete, as discussed in [To20])

9.2. Algorithm parameters

- KDF context length is incorrect, e.g. it does not include TCP header length + payload length (it should)

- KDF calculation does not start from counter i = 1 (it should)

- KDF uses keys generated from current TCP segment sequence numbers (KDF should use only local and remote ISNs or zero, as indicated in Sec 5.2 of TCP-AO [RFC5925])

9.3. String handling issues

The strings indicated in TCP-AO and its algorithms are indicated as a sequence of bytes of known length. In some implementations, string lengths are indicated by a terminal value (e.g., zero in C). This terminal value is not included as part of the string for calculations.

- Password includes the last zero-byte (it should not)

- Label "TCP-AO" includes the last zero byte (it should not)

9.4. Header coverage issues

- TCP checksum and/or MAC is not zeroed properly before calculation (both should be)

- TCP header is not included to the MAC calculation (it should be)

- TCP options are not included to the MAC calculation by default (there is a separate parameter in the master key tuple to ignore options; this document provides test vectors for both options-included and options-excluded cases)

10. Security Considerations

This document is intended to assist in the validation of implementations of TCP-AO, to further enable its more widespread use as a security mechanism to authenticate not only TCP payload contents but the TCP headers and protocol.

The master_key of "testvector" used here for test vector generation SHOULD NOT be used operationally.

11. IANA Considerations

This document contains no IANA issues. This section should be removed upon publication as an RFC.

12. References

12.1. Normative References

[RFC791] Postel, J., "Internet Protocol," RFC 791, Sept. 1981.

[RFC793] Postel, J., "Transmission Control Protocol," RFC 793, September 1981.

[RFC2018] Mathis, M., J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgment Options," RFC 2018, Oct. 1996.

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

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

[RFC5926] Lebovitz, G., and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)," RFC 5925, June 2010.

[RFC6978] Touch, J., "A TCP Authentication Option Extension for NAT Traversal," RFC 6978, July 2013.

[RFC7323] Borman, D., B. Braden, V. Jacobson, R. Scheffenegger, Ed., "TCP Extensions for High Performance," RFC 7323, Sept. 2014.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words," RFC 2119, May 2017.

[RFC8200] Deering, S., R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," RFC 8200, Jul. 2017.

12.2. Informative References

[RFC2202] Cheng, P., and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1," RFC 2202, Sept. 1997.

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

[To20] Touch, J., "Sequence Number Extension for Windowed Protocols," draft-tsvwg-touch-sne, Jun. 2020.

13. Acknowledgments

(TBD)

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Joe Touch
Manhattan Beach, CA 90266 USA
Phone: +1 (310) 560-0334
Email: touch@strayalpha.com

Juhamatti Kuusisaari
Infinera Corporation
Sinimaentie 6c
FI-02630 Espoo, Finland
Email: jkuusisaari@infinera.com