

TCPM
Internet Draft
Intended status: Informational
Expires: June 2022

J. Touch
Independent consultant
J. Kuusisaari
Infinera
December 19, 2021

TCP-A0 Test Vectors

[draft-ietf-tcpm-ao-test-vectors-04.txt](#)

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), 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 June 19, 2022.

Copyright Notice

Copyright (c) 2021 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.

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 middlebox traversal.

Table of Contents

1.	Introduction.....	3
2.	Conventions used in this document.....	4
3.	Input Test Vectors.....	4
3.1.	TCP Connection Parameters.....	4
3.1.1.	TCP-AO parameters.....	4
3.1.2.	Active (client) side parameters.....	4
3.1.3.	Passive (server) side parameters.....	5
3.1.4.	Other IP fields and options.....	5
3.1.5.	Other TCP fields and options.....	5
4.	IPv4 SHA-1 Output Test Vectors.....	5
4.1.	SHA-1 MAC (default - covers TCP options).....	6
4.1.1.	Send (client) SYN (covers options).....	6
4.1.2.	Receive (server) SYN-ACK (covers options).....	6
4.1.3.	Send (client) non-SYN (covers options).....	7
4.1.4.	Receive (server) non-SYN (covers options).....	7
4.2.	SHA-1 MAC (omits TCP options).....	8
4.2.1.	Send (client) SYN (omits options).....	8
4.2.2.	Receive (server) SYN-ACK (omits options).....	8
4.2.3.	Send (client) non-SYN (omits options).....	9
4.2.4.	Receive (server) non-SYN (omits options).....	9
5.	IPv4 AES-128 Output Test Vectors.....	10
5.1.	AES MAC (default - covers TCP options).....	10
5.1.1.	Send (client) SYN (covers options).....	10
5.1.2.	Receive (server) SYN-ACK (covers options).....	11
5.1.3.	Send (client) non-SYN (covers options).....	11
5.1.4.	Receive (server) non-SYN (covers options).....	12
5.2.	AES MAC (omits TCP options).....	12
5.2.1.	Send (client) SYN (omits options).....	12

Touch

Expires June 19, 2022

[Page 2]

5.2.2.	Receive (server) SYN-ACK (omits options).....	13
5.2.3.	Send (client) non-SYN (omits options).....	13
5.2.4.	Receive (server) non-SYN (omits options).....	14
6.	IPv6 SHA-1 Output Test Vectors.....	14
6.1.	SHA-1 MAC (default - covers TCP options).....	15
6.1.1.	Send (client) SYN (covers options).....	15
6.1.2.	Receive (server) SYN-ACK (covers options).....	15
6.1.3.	Send (client) non-SYN (covers options).....	16
6.1.4.	Receive (server) non-SYN (covers options).....	16
6.2.	SHA-1 MAC (omits TCP options).....	17
6.2.1.	Send (client) SYN (omits options).....	17
6.2.2.	Receive (server) SYN-ACK (omits options).....	18
6.2.3.	Send (client) non-SYN (omits options).....	18
6.2.4.	Receive (server) non-SYN (omits options).....	19
7.	IPv6 AES-128 Output Test Vectors.....	19
7.1.	AES MAC (default - covers TCP options).....	19
7.1.1.	Send (client) SYN (covers options).....	19
7.1.2.	Receive (server) SYN-ACK (covers options).....	20
7.1.3.	Send (client) non-SYN (covers options).....	20
7.1.4.	Receive (server) non-SYN (covers options).....	21
7.2.	AES MAC (omits TCP options).....	22
7.2.1.	Send (client) SYN (omits options).....	22
7.2.2.	Receive (server) SYN-ACK (omits options).....	22
7.2.3.	Send (client) non-SYN (omits options).....	23
7.2.4.	Receive (server) non-SYN (omits options).....	23
8.	Observed Implementation Errors.....	24
8.1.	Algorithm issues.....	24
8.2.	Algorithm parameters.....	24
8.3.	String handling issues.....	24
8.4.	Header coverage issues.....	25
9.	Security Considerations.....	25
10.	IANA Considerations.....	25
11.	References.....	25
11.1.	Normative References.....	25
11.2.	Informative References.....	26
12.	Acknowledgments.....	26

1. Introduction

This document provides test vectors to validate the correct implementation of the TCP Authentication Option (TCP-AO) [[RFC5925](#)] and its mandatory cryptographic algorithms defined in [[RFC5926](#)]. 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 for middlebox traversal. It also includes both default key derivation functions (KDFs) and MAC generation

Touch

Expires June 19, 2022

[Page 3]

algorithms [[RFC5926](#)] and lists common pitfalls of implementing the algorithms correctly.

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

This document provides test vectors from multiple implementations that have been validated against each other for interoperability.

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. Input Test Vectors

3.1. 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](#)].

3.1.1. 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 decimal (0x3d hexadecimal)

Passive (server) side KeyID: 84 decimal (0x54 hexadecimal)

Master_key: "testvector" (length = 10 bytes)

3.1.2. 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 (dotted decimal)

For IPv6: fd00::1 (IPv6 hexadecimal)

TCP port: (varies)

3.1.3. 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 (dotted decimal)

For IPv6: fd00::2 (IPv6 hexadecimal)

TCP port = 179 decimal (BGP)

3.1.4. 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 binary (CS7) as is typical for BGP, ECN = 00 binary, set DF, and clear MF.

IPv4 uses a TTL of 255 decimal; IPv6 uses a hop limit of 64 decimal.

All IPv6 packets use the following other parameters [[RFC8200](#)]: traffic class = 0xe0 hexadecimal (DSCP = 111000 binary CS7, as is typical for BGP, with ECN = 00 binary) and no EHs.

3.1.5. 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](#)].

4. IPv4 SHA-1 Output Test Vectors

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

In the following sections, all values are indicated as 2-digit hexadecimal values with spacing per line representing the contents of 16 consecutive bytes, as is typical for data dumps. The IP/TCP data indicates the entire IP packet, including the TCP segment and

its options (whether covered by TCP-A0 or not, as indicated), including TCP-A0.

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

4.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

```
6d 63 ef 1b 02 fe 15 09 d4 b1 40 27 07 fd 7b 04
16 ab b7 4f
```

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
2e e4 37 c6 f8 ed e6 d7 c4 d6 02 e7
```

MAC:

```
2e e4 37 c6 f8 ed e6 d7 c4 d6 02 e7
```

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

Receive_SYN_traffic_key:

```
d9 e2 17 e4 83 4a 80 ca 2f 3f d8 de 2e 41 b8 e6
79 7f ea 96
```

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
ee ab 0f e2 4c 30 10 81 51 16 b3 be
```

MAC:

```
ee ab 0f e2 4c 30 10 81 51 16 b3 be
```

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

Send_other_traffic_key:

```
d2 e5 9c 65 ff c7 b1 a3 93 47 65 64 63 b7 0e dc
24 a1 3d 71
```

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 70 64 cf 99 8c c6 c3 15
c2 c2 e2 bf 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:

```
70 64 cf 99 8c c6 c3 15 c2 c2 e2 bf
```

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

Receive_other_traffic_key:

```
d9 e2 17 e4 83 4a 80 ca 2f 3f d8 de 2e 41 b8 e6
79 7f ea 96
```

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 a6 3f 0e cb bb 2e 63 5c
95 4d ea c7 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:

a6 3f 0e cb bb 2e 63 5c 95 4d ea c7

[4.2.](#) SHA-1 MAC (omits TCP options)

[4.2.1.](#) Send (client) SYN (omits options)

Send_SYN_traffic_key:

30 ea a1 56 0c f0 be 57 da b5 c0 45 22 9f b1 0a
42 3c d7 ea

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
80 af 3c fe b8 53 68 93 7b 8f 9e c2

MAC:

80 af 3c fe b8 53 68 93 7b 8f 9e c2

[4.2.2.](#) Receive (server) SYN-ACK (omits options)

Receive_SYN_traffic_key:

b5 b2 89 6b b3 66 4e 81 76 b0 ed c6 e7 99 52 41a
01 a8 30 7f

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
09 30 6f 9a ce a6 3a 8c 68 cb 9a 70

MAC:

09 30 6f 9a ce a6 3a 8c 68 cb 9a 70

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

Send_other_traffic_key:

f3 db 17 93 d7 91 0e cd 80 6c 34 f1 55 ea 1f 00
34 59 53 e3

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 71 06 08 cc 69 6c 03 a2
71 c9 3a a5 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:

71 06 08 cc 69 6c 03 a2 71 c9 3a a5

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

Receive_other_traffic_key:

b5 b2 89 6b b3 66 4e 81 76 b0 ed c6 e7 99 52 41
01 a8 30 7f

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 97 76 6e 48 ac 26 2d e9
ae 61 b4 f9 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:

```

97 76 6e 48 ac 26 2d e9 ae 61 b4 f9

```

5. IPv4 AES-128 Output Test Vectors

AES-128 is computed as required by TCP-A0 [[RFC5926](#)]

In the following sections, all values are indicated as 2-digit hexadecimal values with spacing per line representing the contents of 16 consecutive bytes, as is typical for data dumps. The IP/TCP data indicates the entire IP packet, including the TCP segment and its options (whether covered by TCP-A0 or not, as indicated), including TCP-A0.

5.1. AES MAC (default - covers TCP options)

5.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

```

f5 b8 b3 d5 f3 4f db b6 eb 8d 4a b9 66 0e 60 e3

```

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
e4 77 e9 9c 80 40 76 54 98 e5 50 91

```

MAC:

e4 77 e9 9c 80 40 76 54 98 e5 50 91

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

Receive_SYN_traffic_key:

4b c7 57 1a 48 6f 32 64 bb d8 88 47 40 66 b4 b1

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
d6 ad a7 bc 4c dd 53 6d 17 69 db 5f

MAC:

d6 ad a7 bc 4c dd 53 6d 17 69 db 5f

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

Send_other_traffic_key:

8c 8a e0 e8 37 1e c5 cb b9 7e a7 9d 90 41 83 91

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 77 41 27 42 fa 4d c4 33
ef f0 97 3e 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:

77 41 27 42 fa 4d c4 33 ef f0 97 3e

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

Receive_other_traffic_key:

4b c7 57 1a 48 6f 32 64 bb d8 88 47 40 66 b4 b1

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 f6 d9 65 a7 83 82 a7 48
45 f7 2d ac 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:

f6 d9 65 a7 83 82 a7 48 45 f7 2d ac

5.2. AES MAC (omits TCP options)

5.2.1. Send (client) SYN (omits options)

Send_SYN_traffic_key:

2c db ae 13 92 c4 94 49 fa 92 c4 50 97 35 d5 0e

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
c4 4e 60 cb 31 f7 c0 b1 de 3d 27 49

MAC:

c4 4e 60 cb 31 f7 c0 b1 de 3d 27 49

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

Receive_SYN_traffic_key:

3c e6 7a 55 18 69 50 6b 63 47 b6 33 c5 0a 62 4a

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
3a 6a bb 20 7e 49 b1 be 71 36 db 90

MAC:

3a 6a bb 20 7e 49 b1 be 71 36 db 90

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

Send_other_traffic_key:

03 5b c4 00 a3 41 ff e5 95 f5 9f 58 00 50 06 ca

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 75 85 e9 e9 d5 c3 ec 85
7b 96 f8 37 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:

75 85 e9 e9 d5 c3 ec 85 7b 96 f8 37

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

Receive_other_traffic_key:

3c e6 7a 55 18 69 50 6b 63 47 b6 33 c5 0a 62 4a

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 5c 04 0f d9 23 33 04 76
5c 09 82 f4 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:

5c 04 0f d9 23 33 04 76 5c 09 82 f4

6. IPv6 SHA-1 Output Test Vectors

SHA-1 is computed as specified for TCP-A0 [[RFC5926](#)].

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

6.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

62 5e c0 9d 57 58 36 ed c9 b6 42 84 18 bb f0 69
89 a3 61 bb

IPv6/TCP:

6e 08 91 dc 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 f7 e4 00 b3 17 6a 83 3f
00 00 00 00 e0 02 ff ff 47 21 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 00 41 d0 87 00 00 00 00
1d 10 3d 54 90 33 ec 3d 73 34 b6 4c 5e dd 03 9f

MAC:

90 33 ec 3d 73 34 b6 4c 5e dd 03 9f

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

Receive_SYN_traffic_key:

e4 a3 7a da 2a 0a fc a8 71 14 34 91 3f e1 38 c7
71 eb cb 4a

IPv6/TCP:

6e 01 00 9e 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 b3 f7 e4 3f 51 99 4b
17 6a 83 40 e0 12 ff ff bf ec 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a bd 33 12 9b 00 41 d0 87
1d 10 54 3d f1 cb a3 46 c3 52 61 63 f7 1f 1f 55

MAC:

f1 cb a3 46 c3 52 61 63 f7 1f 1f 55

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

Send_other_traffic_key:

```
1e d8 29 75 f4 ea 44 4c 61 58 0c 5b d9 0d bd 61
bb c9 1b 7e
```

IPv6/TCP:

```
6e 08 91 dc 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 f7 e4 00 b3 17 6a 83 40
3f 51 99 4c c0 18 01 00 32 9c 00 00 01 01 08 0a
00 41 d0 91 bd 33 12 9b 1d 10 3d 54 bf 08 05 fe
b4 ac 7b 16 3d 6f cd f2 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 79 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 fd
e8 02 08 40 06 00 64 00 01 01 00
```

MAC:

```
bf 08 05 fe b4 ac 7b 16 3d 6f cd f2
```

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

Receive_other_traffic_key:

```
e4 a3 7a da 2a 0a fc a8 71 14 34 91 3f e1 38 c7
71 eb cb 4a
```

IPv6/TCP:

```

6e 01 00 9e 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 00 b3 f7 e4 3f 51 99 4c
17 6a 83 83 c0 18 01 00 ee 6e 00 00 01 01 08 0a
bd 33 12 a5 00 41 d0 91 1d 10 54 3d 6c 48 12 5c
11 33 5b ab 9a 07 a7 97 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 7a 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 fd
e8 02 08 40 06 00 64 00 01 01 00

```

MAC:

```

6c 48 12 5c 11 33 5b ab 9a 07 a7 97

```

[6.2.](#) SHA-1 MAC (omits TCP options)

[6.2.1.](#) Send (client) SYN (omits options)

Send_SYN_traffic_key:

```

31 a3 fa f6 9e ff ae 52 93 1b 7f 84 54 67 31 5c
27 0a 4e dc

```

IPv6/TCP:

```

6e 07 8f cd 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 c6 cd 00 b3 02 0c 1e 69
00 00 00 00 e0 02 ff ff a4 1a 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 00 9d b9 5b 00 00 00 00
1d 10 3d 54 88 56 98 b0 53 0e d4 d5 a1 5f 83 46

```

MAC:

```

88 56 98 b0 53 0e d4 d5 a1 5f 83 46

```

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

Receive_SYN_traffic_key:

```
40 51 08 94 7f 99 65 75 e7 bd bc 26 d4 02 16 a2
c7 fa 91 bd
```

IPv6/TCP:

```
6e 0a 7e 1f 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 00 b3 c6 cd eb a3 73 4d
02 0c 1e 6a e0 12 ff ff 77 4d 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 5e c9 9b 70 00 9d b9 5b
1d 10 54 3d 3c 54 6b ad 97 43 f1 2d f8 b8 01 0d
```

MAC:

```
3c 54 6b ad 97 43 f1 2d f8 b8 01 0d
```

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

Send_other_traffic_key:

```
b3 4e ed 6a 93 96 a6 69 f1 c4 f4 f5 76 18 f3 65
6f 52 c7 ab
```

IPv6/TCP:

```
6e 07 8f cd 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 c6 cd 00 b3 02 0c 1e 6a
eb a3 73 4e c0 18 01 00 83 e6 00 00 01 01 08 0a
00 9d b9 65 5e c9 9b 70 1d 10 3d 54 48 bd 09 3b
19 24 e0 01 19 2f 5b f0 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 79 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 fd
e8 02 08 40 06 00 64 00 01 01 00
```

MAC:

48 bd 09 3b 19 24 e0 01 19 2f 5b f0

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

Receive_other_traffic_key:

40 51 08 94 7f 99 65 75 e7 bd bc 26 d4 02 16 a2
c7 fa 91 bd

IPv6/TCP:

6e 0a 7e 1f 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 b3 c6 cd eb a3 73 4e
02 0c 1e ad c0 18 01 00 71 6a 00 00 01 01 08 0a
5e c9 9b 7a 00 9d b9 65 1d 10 54 3d 55 9a 81 94
45 b4 fd e9 8d 9e 13 17 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 7a 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 fd
e8 02 08 40 06 00 64 00 01 01 00

MAC:

55 9a 81 94 45 b4 fd e9 8d 9e 13 17

7. IPv6 AES-128 Output Test Vectors

AES-128 is computed as required by TCP-A0 [[RFC5926](#)].

7.1. AES MAC (default - covers TCP options)

7.1.1. Send (client) SYN (covers options)

Send_SYN_traffic_key:

fa 5a 21 08 88 2d 39 d0 c7 19 29 17 5a b1 b7 b8

IP/TCP:

```
6e 04 a7 06 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 f8 5a 00 b3 19 3c cc ec
00 00 00 00 e0 02 ff ff de 5d 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 13 e4 ab 99 00 00 00 00
1d 10 3d 54 59 b5 88 10 74 81 ac 6d c3 92 70 40
```

MAC:

```
59 b5 88 10 74 81 ac 6d c3 92 70 40
```

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

Receive_SYN_traffic_key:

```
cf 1b 1e 22 5e 06 a6 36 16 76 4a 06 7b 46 f4 b1
```

IPv6/TCP:

```
6e 06 15 20 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 b3 f8 5a a6 74 4e cb
19 3c cc ed e0 12 ff ff ea bb 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 71 da ab c8 13 e4 ab 99
1d 10 54 3d dc 28 43 a8 4e 78 a6 bc fd c5 ed 80
```

MAC:

```
dc 28 43 a8 4e 78 a6 bc fd c5 ed 80
```

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

Send_other_traffic_key:

```
61 74 c3 55 7a be d2 75 74 db a3 71 85 f0 03 00
```

IPv6/TCP:

```
6e 04 a7 06 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 f8 5a 00 b3 19 3c cc ed
a6 74 4e cc c0 18 01 00 32 80 00 00 01 01 08 0a
13 e4 ab a3 71 da ab c8 1d 10 3d 54 7b 6a 45 5c
0d 4f 5f 01 83 5b aa b3 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 79 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 fd
e8 02 08 40 06 00 64 00 01 01 00
```

MAC:

```
7b 6a 45 5c 0d 4f 5f 01 83 5b aa b3
```

[7.1.4. Receive \(server\) non-SYN \(covers options\)](#)

Receive_other_traffic_key:

```
cf 1b 1e 22 5e 06 a6 36 16 76 4a 06 7b 46 f4 b1
```

IPv6/TCP:

```
6e 06 15 20 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 01 00 b3 f8 5a a6 74 4e cc
19 3c cd 30 c0 18 01 00 52 f4 00 00 01 01 08 0a
71 da ab d3 13 e4 ab a3 1d 10 54 3d c1 06 9b 7d
fd 3d 69 3a 6d f3 f2 89 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 7a 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 fd
e8 02 08 40 06 00 64 00 01 01 00
```

MAC:

```
c1 06 9b 7d fd 3d 69 3a 6d f3 f2 89
```

[7.2.](#) AES MAC (omits TCP options)

[7.2.1.](#) Send (client) SYN (omits options)

Send_SYN_traffic_key:

a9 4f 51 12 63 e4 09 3d 35 dd 81 8c 13 bb bf 53

IPv6/TCP:

6e 09 3d 76 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 f2 88 00 b3 b0 1d a7 4a
00 00 00 00 e0 02 ff ff 75 ff 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 14 27 5b 3b 00 00 00 00
1d 10 3d 54 3d 45 b4 34 2d e8 bb 15 30 84 78 98

MAC:

3d 45 b4 34 2d e8 bb 15 30 84 78 98

[7.2.2.](#) Receive (server) SYN-ACK (omits options)

Receive_SYN_traffic_key:

92 de a5 bb c7 8b 1d 9f 5b 29 52 e9 cd 30 64 2a

IPv6/TCP:

6e 0c 60 0a 00 38 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 b3 f2 88 a6 24 61 45
b0 1d a7 4b e0 12 ff ff a7 0c 00 00 02 04 05 a0
01 03 03 08 04 02 08 0a 17 82 24 5b 14 27 5b 3b
1d 10 54 3d 1d 01 f6 c8 7c 6f 93 ac ff a9 d4 b5

MAC:

1d 01 f6 c8 7c 6f 93 ac ff a9 d4 b5

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

Send_other_traffic_key:

4f b2 08 6e 40 2c 67 90 79 ed 65 d4 bf 97 69 3d

IPv6/TCP:

6e 09 3d 76 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 f2 88 00 b3 b0 1d a7 4b
a6 24 61 46 c0 18 01 00 c3 6d 00 00 01 01 08 0a
14 27 5b 4f 17 82 24 5b 1d 10 3d 54 29 0c f4 14
cc b4 7a 33 32 76 e7 f8 ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 79 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 fd
e8 02 08 40 06 00 64 00 01 01 00

MAC:

29 0c f4 14 cc b4 7a 33 32 76 e7 f8

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

Receive_other_traffic_key:

92 de a5 bb c7 8b 1d 9f 5b 29 52 e9 cd 30 64 2a

IPv6/TCP:

6e 0c 60 0a 00 73 06 40 fd 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 02 fd 00 00 00 00 00 00
00 00 00 00 00 00 00 00 01 00 b3 f2 88 a6 24 61 46
b0 1d a7 8e c0 18 01 00 34 51 00 00 01 01 08 0a
17 82 24 65 14 27 5b 4f 1d 10 54 3d 99 51 5f fc
d5 40 34 99 f6 19 fd 1b ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff 00 43 01 04 fd e8 00 b4
01 01 01 7a 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 fd
e8 02 08 40 06 00 64 00 01 01 00

MAC:

99 51 5f fc d5 40 34 99 f6 19 fd 1b

8. Observed Implementation Errors

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

8.1. Algorithm issues

- o Underlying implementation of HMAC SHA1 or AES128 CMAC does not pass their corresponding test vectors [[RFC2202](#)] [[RFC4493](#)]
- o The SNE algorithm does not consider corner cases (the pseudocode in [[RFC5925](#)] was not intended as complete, as discussed in [[To21](#)], the latter of which includes its own validation sequence)

8.2. Algorithm parameters

- o KDF context length is incorrect, e.g. it does not include TCP header length + payload length (it should, per 5.2 of TCP-AO [[RFC5925](#)])
- o KDF calculation does not start from counter $i = 1$ (it should, per Sec. 3.1.1 of the TCP-AO crypto algorithms [[RFC5926](#)])
- o KDF calculation does not include output length in bits, contained in two bytes in network byte order (it should, per Sec. 3.1.1 of the TCP-AO crypto algorithms [[RFC5926](#)])
- o 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](#)])

8.3. 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.

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

8.4. Header coverage issues

- o TCP checksum and/or MAC is not zeroed properly before calculation (both should be)
- o TCP header is not included to the MAC calculation (it should be)
- o 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)

9. 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.

10. IANA Considerations

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

11. References

11.1. 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 5926](#), 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 8174](#), May 2017.
- [RFC8200] Deering, S., R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," [RFC 8200](#), Jul. 2017.

[11.2](#). 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.
- [To21] Touch, J., "Sequence Number Extension for Windowed Protocols," [draft-touch-sne](#), Nov. 2021.

[12](#). Acknowledgments

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

