TCP Authentication Option

With TCP Authentication Option (TCP-AO), defined in RFC 5925, you can protect long-lived TCP connections against replays using stronger Message Authentication Codes (MACs).

Overview of TCP Authentication Option

TCP-AO is the proposed replacement for TCP MD5, defined in RFC 2385. Unlike TCP MD5, TCP-AO is resistant to collision attacks and provides algorithmic agility and support for key management.

TCP-AO has the following distinct features:

- TCP-AO supports the use of stronger Message Authentication Codes (MACs) to enhance the security of long-lived TCP connections.
- TCP-AO protects against replays for long-lived TCP connections, and coordinates key changes between endpoints by providing a more explicit key management.

TCP-AO is supported along with TCP MD5, and you can choose one of the authentication methods. However, a configuration in which one of the devices is configured with the TCP MD5 option and the other with the TCP-AO option is not supported.

TCP-AO Key Chain

TCP-AO is based on traffic keys and Message Authentication Codes (MACs) generated using the keys and a MAC algorithm. The traffic keys are derived from primary keys that you can configure in a TCP-AO key chain. Use the **key chain** *key-chain-name* **tcp** command in the global configuration mode to create a TCP-AO key chain and configure keys in the chain. The TCP-AO key chain must be configured on both the peers communicating via a TCP connection.

Keys in a TCP-AO key chain have the following configurable properties:

Configurable Property	Description
send-id	Key identifier of the TCP-AO option of the outgoing segment. The send identifier configured on a router must match the receive identifier configured on the peer.
recv-id	Key identifier compared with the TCP-AO key identifier of the incoming segment during authentication. The receive identifier configured on a router must match the send identifier configured on the peer.
cryptographic-algorithm	The MAC algorithm to be used to create MACs for outgoing segments. The algorithm can be one of the following: • AES-128-CMAC authentication algorithm • HMAC-SHA-1 authentication algorithm • HMAC-SHA-256 authentication algorithm.
include-tcp-options	This flag indicates whether TCP options other than TCP-AO will be used to calculate MACs. With this flag enabled, the contents of all options along with a zero-filled authentication option, is used to calculate the MAC. When the flag is disabled, all options other than TCP-AO are excluded from MAC calculations. This flag is disabled by default. **Note** The configuration of this flag is overridden by the application configuration when the application configuration is available.
send-lifetime	This configuration determines the time for which a key is valid and can be used for TCP-AO-based authentication of TCP segments to be sent. When the lifetime of key elapses and the key expires, the next key with the longest lifetime is selected.
accept-lifetime	This configuration determines the time for which a key is valid and can be used for TCP-AO-based authentication of received TCP segments.
key-string	The key string is a pre-shared primary key configured on both peers and is used to derive the traffic keys.

Configurable Property	Description
accept-ao-mismatch	This flag determines whether the receiver accepts segments for which the MAC in the incoming TCP-AO does not match the MAC generated on the receiver. With this configuration, incoming segments without TCP Authentication Option are also accepted. **Note** • Use this configuration with caution. This configuration disables TCP-AO functionality and key rollover on associated connections. • The configuration of this flag is overridden by the application configuration when the application configuration is available.

Primary Key Tuples

The key chain and keys are used to create Primary Key Tuples that are optimized for look-ups during TCP send and receive operations. The Primary Key Tuples consists of a primary key, identifiers for the key, algorithms to be used for the Key Derivation Function (KDF) and MAC, and other properties.

On both the peers, two pointers called current-key and next-key are used to track Primary Key Tuples .

- current-key: Identifies the Primary Key Tuples that is being used to compute traffic keys for outgoing TCP segments.
- next-key: Identifies the Primary Key Tuples that is ready to be used to authenticate received segments.

Traffic Keys

Traffic keys are used to compute MACs of segment data using an MAC algorithm. Traffic keys are derived using a Key Derivation Function (KDF) from an Primary Key Tuples and the KDF context. The KDF context consists of the local and remote IP address pairs and TCP port numbers. For established connections, the KDF context also includes the TCP Initial Sequence Numbers (ISNs) in each direction.

A single Primary Key Tuple can be used to derive the four traffic keys in the following list. An endpoint uses at least three of the keys for authentication.

- Send SYN Traffic Key – the traffic key used to authenticate outgoing SYNs.
- Receive SYN Traffic Key – the traffic key used to authenticate incoming SYNs.
- Send Other Key – the traffic key used to authenticate all other outgoing TCP segments.
- Receive Other Key – the traffic key used to authenticate all other incoming TCP segments.

Message Authentication Codes

An MAC is computed for a TCP segment using the configured MAC algorithm, relevant traffic keys, and the TCP segment data prefixed with a pseudo-header.

Protection from Replays in Long-lived TCP Connections

The 32-bit sequence number of TCP segments may roll over and repeat in the case of long-lived TCP connections. As a result of a repetition of sequence numbers, TCP Segments may get replayed within a connection. To avoid this, TCP-AO uses a 32-bit Sequence Number Extension (SNE) in the pseudo-header along with the TCP sequence number for transmitted and received segments. Thus, TCP-AO emulates a 64-bit sequence number space by combining SNE and the TCP sequence number.

TCP-AO Format

TCP-AO has the following TLV format in the options sequence of a TCP segment:

Kind (1B) = 29	Length (1B)	KeyID (1B)	RNextKeyID (1B)
MAC (12-16B)			
MAC			
MAC			
MAC			

The fields of the TLV format are as follows:

- Kind: Indicates TCP-AO with a value of 29.
- Length: Indicates the length of the TCP-AO sequence.
- KeyID: The send identifier of the Primary Key Tuples that was used to generate the traffic keys.
- RNextKeyID: The receive identifier of the Primary Key Tuples that is ready to be used to authenticate received segments.
- MAC: The MAC computed for the TCP segment data and the prefixed pseudo header.

TCP-AO Key Rollover

TCP-AO keys are valid for a defined duration configured using the send-lifetime and accept-lifetime properties. If send-lifetime and accept-lifetime are not configured for a key, the key has infinite send and accept lifetimes. Key rollover is initiated based on the send lifetimes of keys. As part of key rollover, a key that is valid and has the longest send lifetime into the future is selected as the active key.

When key rollover is initiated, one of the peer routers, say Router A, indicates that the rollover is necessary. To indicate that the rollover is necessary, Router A sets the RNextKeyID to the receive identifier of the new Primary Key Tuples to be used. On receiving the TCP segment, the peer router, say Router B, finds the Primary Key Tuples indicated by the RNextKeyID in the TCP-AO payload. If the key is available and valid, Router B sets the current key to the new Primary Key Tuples. After Router B has rolled over, Router A also sets the current key to the new Primary Key Tuples.

Key rollover can be initiated by one of the following methods:

- Rollover on send-lifetime expiry
- Rollover with overlapping send-lifetimes

If you do not configure a new key that can be activated before the expiry of the current key, the key may time out and expire. Such an expiry can cause retransmissions with the peer router rejecting segments authenticated with the expired key. The connection may fail due to Retransmission Time Out (RTO). When new valid keys are configured, a new connection is established.

Note

- Key rollover is based only on send lifetimes of keys.
- Key rollover is only supported within a key chain.
- Forced deletion of a key in use does not trigger key rollover.
- From among the keys in a key chain, the key with the longest send lifetime into the future is selected as the active key during a rollover.

Restrictions for TCP Authentication Option

- The send-id and recv-id of each key in the key chain must be unique. Because send-id and recv-id must be chosen from the range 0 to 255, the TCP-AO key chain can have a maximum of 256 keys.
- Only one keychain can be associated with an application connection. Rollover is always performed within the keys in this keychain.
- TCP-AO does not allow the modification of a key in use. Modify a key after disassociating the key from the connection.
- If the key in use expires, expect segment loss until a new key that has a valid lifetime is configured on each side and keys rollover.

How to Configure TCP Authentication Option

Configure TCP Key Chain and Keys

Configure TCP-AO key chain and keys on both the peers communicating through a TCP connection.

Note

- Ensure that the key-string, send-lifetimes, cryptographic-algorithm, and ids of keys match on both peers.
- Ensure that the send-id on a router matches the recv-id on the peer router. We recommend using the same id for both the parameters unless there is a need to use separate key spaces.
- The send-id and recv-id of a key cannot be reused for another key in the same key chain.
- Do not modify properties of a key in use, except when you need to modify the send-lifetime of the key to trigger rollover. Before modifying properties other than send-lifetime, disassociate the key from the TCP connection.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode. Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **key chain** *key-chain-name* **tcp**

Example:

```
Device(config)# key chain kc1 tcp
```

Creates a TCP-AO key chain of with a specified name and enters the TCP-AO key chain configuration mode.

The key chain name can have a maximum of 256 characters.

Step 4 **key** *key-id*

Example:

```
Device(config-keychain-tcp)# key 10
```

Creates a key with the specified key-id and enters the TCP-AO key chain key configuration mode.

The key-id must be in the range from 0 to 2147483647.

Note The key-id has only local significance. It is not part of the TCP Authentication Option.

Step 5 **send-id** *send-identifier*

Example:

```
Device(config-keychain-tcp-key)# send-id 218
```

Specifies the send identifier for the key.

The send-identifier must be in the range from 0 to 255.

Step 6 **recv-id** *receiver-identifier*

Example:

```
Device(config-keychain-tcp-key)# recv-id 218
```

Specifies the receive identifier for the key.

The receive-identifier must be in the range from 0 to 255.

Step 7 **cryptographic-algorithm** {**aes-128-cmac** | **hmac-sha-1** | **hmac-sha-256**}

Example:

```
Device(config-keychain-tcp-key)# cryptographic-algorithm hmac-sha-1
```

Specifies the algorithm to be used to compute MACs for TCP segments.

aes-128-cmac	AES-128-CMAC-96: Configures AES-128-CMAC as a cryptographic algorithm with a digest size of 12 bytes.
hmac-sha-1	HMAC-SHA1-96: Configures HMAC-SHA1-96 as a cryptographic algorithm with a digest size of 12 bytes.
hmac-sha-256	HMAC-SHA-256: Configures HMAC-SHA-256 as a cryptographic algorithm with a digest size of 32 bytes.

Step 8 (Optional) **include-tcp-options**

Example:

```
Device(config-keychain-tcp-key)# include-tcp-options
```

This flag indicates whether TCP options other than TCP-AO must be used to calculate MACs.

With the flag enabled, the content of all options, in the order present, is included in the MAC and TCP-AO's MAC field is zero-filled.

When the flag is disabled, all options other than TCP-AO are excluded from MAC calculations.

By default, this flag is disabled.

Step 9 **send-lifetime** [**local**] *start-time* {**infinite** | *end-time* | **duration** *seconds*}

Example:

```
Device(config-keychain-tcp-key)# send-lifetime local 12:00:00 28 Feb 2018 duration 20
```

Specifies the time for which the key is valid to be used for TCP-AO authentication in the send direction.

Use the **local** keyword to specify the start-time in the local time zone. By default, the start-time corresponds to UTC time.

Step 10 **key-string** *master-key*

Example:

```
Device(config-keychain-tcp-key)# key-string abcde
```

Specifies the primary-key for deriving traffic keys.

The primary-keys must be identical on both the peers. If the primary-keys do not match, authentication fails and segments may be rejected by the receiver.

Step 11 (Optional) **accept-ao-mismatch**

Example:

```
Device(config-keychain-tcp-key)# accept-ao-mismatch
```

This flag indicates whether the receiver should accept segments for which the MAC in the incoming TCP AO does not match the MAC generated on the receiver.

Note Use this configuration with caution. This configuration disables TCP-AO functionality and key rollover on associated connections.

Step 12 **end**

Example:

```
Device(config-keychain-tcp-key)# end
```

Exits TCP-AO key chain key configuration mode and returns to privileged EXEC mode.

Verifying TCP-AO Key Chain and Key Configuration

Use the **show key chain** *key-chain-name* command in the privileged EXEC mode to display information about a TCP-AO key chain and keys, and association with TCBs.

```
Router# show key chain key-chain-name
```

```
Router1# show key chain kc1
Key-chain kc1:
    TCP key chain
    key 7893 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (12:32:00 IST Nov 9 2018) - (10:30:00 IST Dec 30 2019) [valid now]
        send lifetime (13:05:00 IST Jan 12 2019) - (10:31:00 IST Dec 30 2019) [valid now]
        send-id - 218
        recv-id - 218
        include-tcp-options
        MKT ready - true
        MKT preferred - true
        MKT in-use - true
        MKT id - 7893
        MKT send-id - 218
        MKT recv-id - 218
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - true
        MKT accept AO mismatch - false
            TCB - 0x7FBD68361838
            curr key - 7893
            next key - 7893
```

Verifying TCP-AO Key Chain Information in the TCB

Use the **show tcp tcb** *address-of-tcb* command in the privileged EXEC mode to display information about TCP-AO in the Transmission Control Block. Obtain *address-of-tcb*(the hexadecimal address of the TCB) from the output of the **show key chain** *key-chain-name* command.

```
Router# show tcp tcb address-of-tcb
```

```
Router1# show tcp tcb 7FBD68361838
Connection state is ESTAB, I/O status: 1, unread input bytes: 0
Connection is ECN Disabled, Minimum incoming TTL 0, Outgoing TTL 255
Local host: 1.0.2.1, Local port: 40125
Foreign host: 1.0.2.2, Foreign port: 5555
Connection tableid (VRF): 0
Maximum output segment queue size: 50

Enqueued packets for retransmit: 0, input: 0  mis-ordered: 0 (0 bytes)

Event Timers (current time is 0x2818B07):
Timer          Starts    Wakeups            Next
Retrans             1          0             0x0
TimeWait            0          0             0x0
```

```
AckHold             1          0             0x0
SendWnd             0          0             0x0
KeepAlive        6651          0       0x281AC36
GiveUp              0          0             0x0
PmtuAger            0          0             0x0
DeadWait            0          0             0x0
Linger              0          0             0x0
ProcessQ            0          0             0x0

iss: 3307331702  snduna: 3307331703  sndnxt: 3307331703
irs:  725047078  rcvnxt:  725047079

sndwnd:   4128  scale:      0  maxrcvwnd:   4128
rcvwnd:   4128  scale:      0  delrcvwnd:      0

SRTT: 125 ms, RTTO: 2625 ms, RTV: 2500 ms, KRTT: 0 ms
minRTT: 15 ms, maxRTT: 1000 ms, ACK hold: 200 ms
uptime: 40996359 ms, Sent idletime: 6505 ms, Receive idletime: 6505 ms
Status Flags: active open
Option Flags: keepalive running, nagle, Retrans timeout
IP Precedence value : 0

TCP AO Key chain: kc1

TCP AO Current Key:
 Id: 7893, Send-Id: 218, Recv-Id: 218
 Include TCP Options: Yes*
 Accept AO Mismatch: No*

TCP AO Next Key:
 Id: 7893, Send-Id: 218, Recv-Id: 218
 Include TCP Options: Yes*
 Accept AO Mismatch: No*

Datagrams (max data segment is 1460 bytes):
Rcvd: 4372 (out of order: 0), with data: 0, total data bytes: 0
Sent: 4372 (retransmit: 0, fastretransmit: 0, partialack: 0, Second Congestion: 0), with
data: 0, total data bytes: 0

 Packets received in fast path: 0, fast processed: 0, slow path: 0
 fast lock acquisition failures: 0, slow path: 0
TCP Semaphore      0x7FBD6801B2E0  FREE

* - Derived from Key
```

Configuring Key Rollover on Send Lifetime Expiry

Configure a new key in the key chain such that the key becomes active on the expiry of the send-lifetime of the currently active key. The examples in the following steps show sample configurations on two peer routers, Router 1 and Router 2. In these examples, the active key has an id of 7890 and the new key has an id of 7891.

Step 1 Identify the active key on both peer routers.

Example:

Identify active key on Router 1:

```
Router1#show run | sec key
key chain kc1 tcp
 key 7890
```

```
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
```

Identify active key on Router 2:

```
Router2# show run | sec key
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
```

Step 2 Configure the new key on both peer routers.

Example:

Configure new key on Router 1:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
 key 7891
  send-id 216
  recv-id 216
  cryptographic-algorithm hmac-sha-1
  key-string fghij
```

Configure new key on Router 2:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
 key 7891
  send-id 216
  recv-id 216
  cryptographic-algorithm hmac-sha-1
  key-string fghij
```

When the send-lifetime of the active key expires, the new key is activated. Syslog messages are displayed indicating rollover to the new key.

Step 3 Reduce the send-lifetimes of active keys on the peer routers.

Example:

Reduce send-lifetime of the active key on Router 1:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
   send-lifetime local 10:00:00 Jun 24 2019 13:45:00 Jun 24 2019
 key 7891
  send-id 216
  recv-id 216
```

```
  cryptographic-algorithm hmac-sha-1
  key-string fghij
```

Reduce send-lifetime of active key on Router 2:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
   send-lifetime local 10:00:00 Jun 24 2019 13:45:00 Jun 24 2019
 key 7891
  send-id 216
  recv-id 216
  cryptographic-algorithm hmac-sha-1
  key-string fghij
```

Step 4 Verify the send-lifetimes of the currently active and new keys on the peer routers.

Example:

Verify send-lifetimes of the keys on Router 1:

```
Router1# sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (13:45:00 IST Jun 24 2019) --- [valid now]
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - false
        MKT in-use - true
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
            TCB - 0x7FC0EC097AC0
            curr key - 7890
            next key - 7890
            TCB - 0x7FC0EBBE7600
            curr key - 7890
            next key - 7890
    key 7891 -- text ”fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) --- [valid now]
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - true
        MKT in-use - false
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - true
        MKT alive (recv) - true
```

```
        MKT include TCP options - false
        MKT accept AO mismatch - false
```

Verify send-lifetimes of the keys on Router 2:

```
Router2# sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (13:45:00 IST Jun 24 2019) --- [valid now]
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - false
        MKT in-use - true
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
            TCB - 0x7FB6BEF4CC10
            curr key - 7890
            next key - 7890
            TCB - 0x7FB6BEAA7B28
            curr key - 7890
            next key - 7890
    key 7891 -- text ”fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) --- [valid now]
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - true
        MKT in-use - false
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
```

Step 5 Verify key rollover on the routers using the **show key chain** command.

Example:

Verify key rollover on Router 1:

```
Router1#
*Jun 24 08:15:00.000: %TCP-6-AOKEYSENDEXPIRED: TCP AO Keychain kc1 key 7890 send lifetime expired
*Jun 24 08:15:00.000: %TCP-6-AOROLLOVER: TCP AO Keychain kc1 rollover from key 7890 to key 7891

Router1#sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
```

```
        send lifetime (10:00:00 IST Jun 24 2019) - (13:45:00 IST Jun 24 2019)
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - false
        MKT in-use - false
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - false
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
    key 7891 -- text ”fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (always valid) - (always valid) [valid now]
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - true
        MKT in-use - true
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
            TCB - 0x7FC0EBBE7600
            curr key - 7891
            next key - 7891
            TCB - 0x7FC0EC097AC0
            curr key - 7891
            next key - 7891
```

Verify key rollover on Router 2:

```
Router2#
*Jun 24 08:15:00.000: %TCP-6-AOKEYSENDEXPIRED: TCP AO Keychain kc1 key 7890 send lifetime expired
*Jun 24 08:15:00.000: %TCP-6-AOROLLOVER: TCP AO Keychain kc1 rollover from key 7890 to key 7891

Router2#sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (13:45:00 IST Jun 24 2019)
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - false
        MKT in-use - false
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - false
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
    key 7891 -- text ”fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
```

```
send lifetime (always valid) - (always valid) [valid now]
send-id - 216
recv-id - 216
MKT ready - true
MKT preferred - true
MKT in-use - true
MKT id - 7891
MKT send-id - 216
MKT recv-id - 216
MKT alive (send) - true
MKT alive (recv) - true
MKT include TCP options - false
MKT accept AO mismatch - false
    TCB - 0x7FB6BEAA7B28
    curr key - 7891
    next key - 7891
    TCB - 0x7FB6BEF4CC10
    curr key - 7891
    next key - 7891
```

Configuring Key Rollover with Overlapping Send Lifetimes

Configure a new key in the key chain such that the currently active key and new key have overlapping send-lifetime values. Also, configure the send-lifetime of the new key such that it extends longer into the future than the send-lifetime of the currently active key. During key rollover, the key with the longest send-lifetime into the future is selected as the active key. Thus, when the send-lifetime of the new key begins, the key becomes active.

The examples in the following steps show sample configurations on two peer routers, Router 1 and Router 2. In these examples, the active key has an id of 7890 and the new key has an id of 7891.

Step 1 Identify the active key on both peer routers.

Example:

Identify active key on Router 1:

```
Router1# show run | sec key
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
  send-lifetime local 10:00:00 Jun 24 2019
   10:00:00 Aug 24 2019
```

Identify active key on Router 2:

```
Router2# show run | sec key
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
  send-lifetime local 10:00:00 Jun 24 2019
   10:00:00 Aug 24 2019
```

Step 2 Configure a new key with an overlapping send-lifetime on both peer routers.

Example:

Configure new key on Router 1:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
  send-lifetime local 10:00:00 Jun 24 2019 10:00:00 Aug 24 2019
 key 7891
  send-id 216
  recv-id 216
  cryptographic-algorithm hmac-sha-1
  key-string fghij
  send-lifetime local 21:50:00 Jun 24 2019 11:00:00 Aug 24 2019
```

Configure new key on Router 2:

```
key chain kc1 tcp
 key 7890
  send-id 215
  recv-id 215
  cryptographic-algorithm hmac-sha-1
  key-string abcde
  send-lifetime local 10:00:00 Jun 24 2019 10:00:00 Aug 24 2019
 key 7891
  send-id 216
  recv-id 216
  cryptographic-algorithm hmac-sha-1
  key-string fghij
  send-lifetime local 21:50:00 Jun 24 2019 11:00:00 Aug 24 2019
```

When the send-lifetime of the new key starts, the new key is activated. Syslog messages are displayed indicating rollover to the new key.

Step 3 Verify that the send-lifetimes of the currently active and new keys are overlapping.

Example:

Verify send-lifetimes of the keys on Router 1:

```
Router1# sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7890
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (10:00:00 IST Aug 24 2019)--- [valid now]
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - true
        MKT in-use - true
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
```

```
            TCB - 0x7F8352155318
            curr key - 7890
            next key - 7890
            TCB - 0x7F8352FF37F0
            curr key - 7890
            next key - 7890
    key 7891 -- text "fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (21:50:00 IST Jun 24 2019) - (11:00:00 IST Aug 24 2019)
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - false
        MKT in-use - false
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - false
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
```

Verify send-lifetimes of the keys on Router 2:

```
Router2#sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7890
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (10:00:00 IST Aug 24 2019)--- [valid now]
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - true
        MKT in-use - true
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
            TCB - 0x7F5FCD185150
            curr key - 7890
            next key - 7890
            TCB - 0x7F5FD2734C48
            curr key - 7890
            next key - 7890
    key 7891 -- text "fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (21:50:00 IST Jun 24 2019) - (11:00:00 IST Aug 24 2019)
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - false
        MKT in-use - false
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - false
        MKT alive (recv) - true
```

```
        MKT include TCP options - false
        MKT accept AO mismatch – false
```

Step 4 Verify key rollover on the routers using the **show key chain** command.

Example:

Verify key rollover on Router 1:

```
Router1#
*Jun 24 16:20:00.000: %TCP-6-AOROLLOVER: TCP AO Keychain kc1 rollover from key 7890 to key 7891
Router1#sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (10:00:00 IST Jun 24 2019) - (10:00:00 IST Aug 24 2019) [valid now]
        send-id - 215
        recv-id - 215
        MKT ready - true
        MKT preferred - false
        MKT in-use - false
        MKT id - 7890
        MKT send-id - 215
        MKT recv-id - 215
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
    key 7891 -- text "fghij"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
        send lifetime (21:50:00 IST Jun 24 2019) - (11:00:00 IST Aug 24 2019) [valid now]
        send-id - 216
        recv-id - 216
        MKT ready - true
        MKT preferred - true
        MKT in-use - true
        MKT id - 7891
        MKT send-id - 216
        MKT recv-id - 216
        MKT alive (send) - true
        MKT alive (recv) - true
        MKT include TCP options - false
        MKT accept AO mismatch - false
            TCB - 0x7F8352FF37F0
            curr key - 7891
            next key - 7891
            TCB - 0x7F8352155318
            curr key - 7891
            next key - 7891
```

Verify key rollover on Router 2:

```
Router2#
*Jun 24 16:20:00.000: %TCP-6-AOROLLOVER: TCP AO Keychain kc1 rollover from key 7890 to key 7891
Router2#sh key chain
Key-chain kc1:
    TCP key chain
    Preferred MKT id - 7891
    key 7890 -- text "abcde"
        cryptographic-algorithm: hmac-sha-1
        accept lifetime (always valid) - (always valid) [valid now]
```

```
    send lifetime (10:00:00 IST Jun 24 2019) - (10:00:00 IST Aug 24 2019) [valid now]
    send-id - 215
    recv-id - 215
    MKT ready - true
    MKT preferred - false
    MKT in-use - false
    MKT id - 7890
    MKT send-id - 215
    MKT recv-id - 215
    MKT alive (send) - true
    MKT alive (recv) - true
    MKT include TCP options - false
    MKT accept AO mismatch - false
key 7891 -- text "fghij"
    cryptographic-algorithm: hmac-sha-1
    accept lifetime (always valid) - (always valid) [valid now]
    send lifetime (21:50:00 IST Jun 24 2019) - (11:00:00 IST Aug 24 2019) [valid now]
    send-id - 216
    recv-id - 216
    MKT ready - true
    MKT preferred - true
    MKT in-use - true
    MKT id - 7891
    MKT send-id - 216
    MKT recv-id - 216
    MKT alive (send) - true
    MKT alive (recv) - true
    MKT include TCP options - false
    MKT accept AO mismatch - false
        TCB - 0x7F5FD2734C48
        curr key - 7891
        next key - 7891
        TCB - 0x7F5FCD185150
        curr key - 7891
        next key - 7891
```

Feature Information for TCP Authentication Option

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1: Feature Information for TCP Authentication Option