# Understanding MACSec Encryption

Security breaches can occur at any layer of the OSI model. At Layer 2, some of the common breaches are MAC address spoofing, ARP spoofing, Denial of Service (DoS) attacks against a DHCP server, and VLAN hopping.

MACSec secures data on physical media, making it impossible for data to be compromised at higher layers. As a result, MACSec encryption takes priority over any other encryption method such as IPsec and SSL at higher layers. MACSec is configured on the Customer Edge (CE) router interfaces that connect to Provider Edge (PE) routers and on all the provider router interfaces.

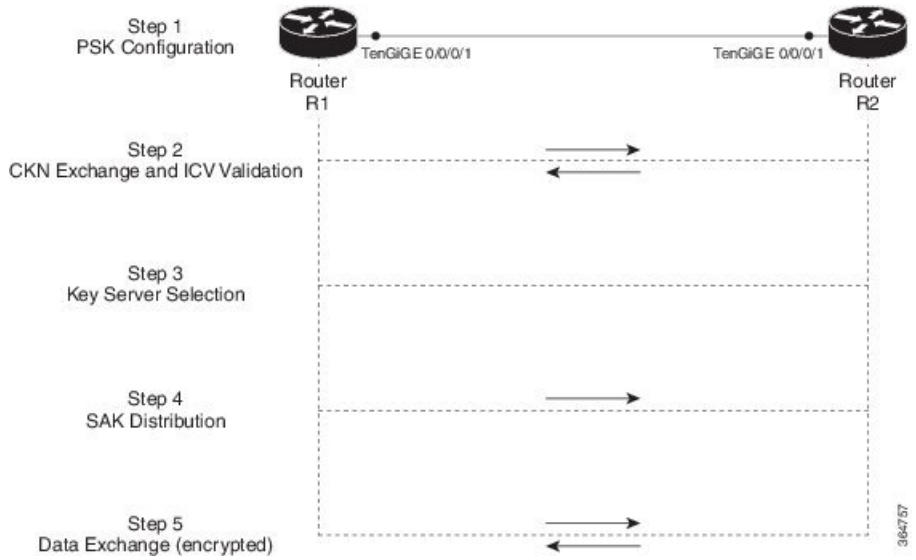

- MKA Authentication Process, on page 1
- MACsec Frame Format, on page 2
- Advantages of Using MACsec Encryption, on page 3
- Hardware Support for MACSec, on page 3
- MACsec PSK, on page 5
- Fallback PSK, on page 5
- Configuring and Verifying MACsec Encryption , on page 6
- Creating a MACsec Keychain, on page 6
- Creating a User-Defined MACsec Policy, on page 9
- Applying MACsec Configuration on an Interface, on page 12
- MACsec Policy Exceptions, on page 13
- Verifying MACsec Encryption on IOS XR, on page 14
- Verifying MACsec Encryption on the Router, on page 21
- Quantum safe key distribution options for MACsec, on page 24
- Secure Key Integration Protocol, on page 32


# MKA Authentication Process

MACsec provides the secure MAC Service on a frame-by-frame basis, using GCM-AES algorithm. MACsec uses the MACsec Key Agreement protocol (MKA) to exchange session keys, and manage encryption keys.

The MACsec encryption process is illustrated in the following figure and description.

*Figure 1: MKA Encryption Process*




**Step 1**: When a link is first established between two routers, they become peers. Mutual peer authentication takes place by configuring a Pre-shared Key (PSK).

**Step 2**: On successful peer authentication, a connectivity association is formed between the peers, and a secure Connectivity Association Key Name (CKN) is exchanged. After the exchange, the MKA ICV is validated with a Connectivity Association Key (CAK), which is effectively a secret key.

**Step 3**: A key server is selected between the routers, based on the configured key server priority. Lower the priority value, higher the preference for the router to become the key server. If no value is configured, the default value of 16 is taken to be the key server priority value for the router. Lowest priority value configures that router as the key server, while the other router functions as a key client. The following rules apply to key server selection:

- Numerically lower values of key server priority and SCI are accorded the highest preference.
- Each router selects a peer advertising the highest preference as its key server provided that peer has not selected another router as its key server or is not willing to function as the key server.
- In the event of a tie for highest preferred key server, the router with the highest priority SCI is chosen as key server (KS).

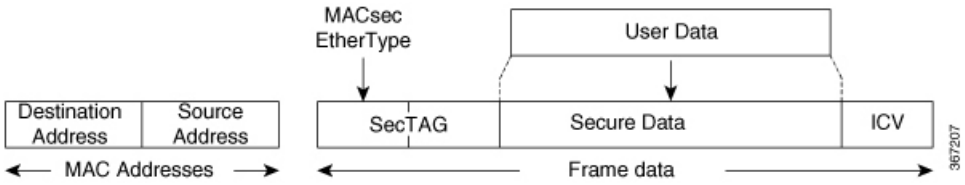
**Step 4**: A security association is formed between the peers. The key server generates and distributes the Secure Association Key (SAK) to the key client (peer). Each secure channel is supported by an overlapped sequence of Security Associations (SA). Each SA uses a new Secure Association Key (SAK).

**Step 5**: Encrypted data is exchanged between the peers.

# MACsec Frame Format

The MACsec header in a frame consists of three components as illustrated in the following figure.

*Figure 2: MACsec Frame Format*




- **SecTAG**: The security tag is 8-16 bytes in length and identifies the SAK to be used for the frame. With Secure Channel Identifier (SCI) encoding, the security tag is 16 bytes in length, and without the encoding, 8 bytes in length (SCI encoding is optional).The security tag also provides replay protection when frames are received out of sequence.
- **Secure Data**: This is the data in the frame that is encrypted using MACsec and can be 2 or more octets in length.
- **ICV**: The ICV provides the integrity check for the frame and is usually 8-16 bytes in length, depending on the cipher suite. Frames that do not match the expected ICV are dropped at the port.

# Advantages of Using MACsec Encryption

- **Data Integrity Check**: Integrity check value (ICV) is used to perform integrity check. The ICV is sent with the protected data unit and is recalculated and compared by the receiver to detect data modification.
- **Data Encryption**: Enables a port to encrypt outbound frames and decrypt MACsec-encrypted inbound frames.
- **Replay Protection**: When frames are transmitted through the network, there is a strong possibility of frames getting out of the ordered sequence. MACsec provides a configurable window that accepts a specified number of out-of-sequence frames.
- **Support for Clear Traffic**: If configured accordingly, data that is not encrypted is allowed to transit through the port.

# Hardware Support for MACSec

**Note** MACSec is supported only on the QSFP28 or SFP28 ports (in 10G to 100G speeds, and not 1G speed).

## Release History for MACSec Hardware Support

*Table 1: Release History for MACSec Hardware Support*

| Release | Description |
|---|---|
| Release 24.1.1 | We now support MACsec for the 1GbE optical SFP transceivers (select variants only*) by encrypting Ethernet frames at the link layer to secure communication for all traffic in Ethernet-based networks.<br>This feature is supported on:<br>• N540-ACC-SYS<br>• N540X-ACC-SYS<br>• N540-24Z8Q2C-SYS<br>• SFP-1G-SX (optical SFP)<br>• SFP-1G-LH (optical SFP)<br>**Note**<br>We recommend the following port numbers to be configured as 10GbE:<br>• 24-27<br>• 28-31 |
| Release 7.7.1 | MACSec, the Layer 2 encryption protocol, secures the data on physical media and provides data integrity and confidentiality.<br>This release introduces the support for MACSec on the following NCS 540 router variant:<br>• N540-24Q8L2DD-SYS<br>**Note**<br>On the N540-24Q8L2DD-SYS router, MACSec is supported on 10G, 25G, 40G, 50G ,100G, 400G, 4x10G, 4x25G, 4x100G, and 2x100G on ports 0 to 9.<br>Ports 0 and 1 are QDD (400G) ports.<br>Ports 2 to 9 are SFP/SFP+ ports.<br>**Note**<br>Data delay protection (DDP) is not supported on Cisco N540-24Q8L2DD-SYS routers. |

| Release | Description |
|---|---|
| Release 7.5.1 | This release introduces the support for MACSec on the following NCS 540 router variant:<br>• N540X-16Z4G8Q2C-A/D |
| Release 7.3.1 | This release introduces the support for MACSec on the following NCS 540 router variants:<br>• N540-ACC-SYS<br>• N540-24Z8Q2C-SYS |
| Release 7.3.1 | MACsec is supported on SFP28 and QSFP28 ports on the following NCS540 variants:<br>• N540-ACC-SYS<br>• N540X-ACC-SYS (Premium)<br>• N540-24Z8Q2C-SYS |

# MACsec PSK

A pre-shared key includes a connectivity association key name (CKN) and a connectivity association key (CAK). A pre-shared key is exchanged between two devices at each end of a point-to-point link to enable MACsec using static CAK security mode. The MACsec Key Agreement (MKA) protocol is enabled after the pre-shared keys are successfully verified and exchanged. The pre-shared keys, the CKN and CAK, must match on both ends of a link.

For more information on MACsec PSK configuration, see Step 3, on page 12 of the Applying MACsec Configuration on an Interface, on page 12 section.

# Fallback PSK

Fallback is a session recovery mechanism when primary PSK fails to bring up secured MKA session. It ensures that a PSK is always available to perform MACsec encryption and decryption.

- In CAK rollover of primary keys, if latest active keys are mismatched, system performs a hitless rollover from current active key to fallback key, provided the fallback keys match.
- If a session is up with fallback, and primary latest active key configuration mismatches are rectified between peers, system performs a hitless rollover from fallback to primary latest active key.

**Note** A valid Fallback PSK (CKN and CAK) must be configured with infinite lifetime. If the fallback PSK is configured with CAK mismatch, the only recovery mechanism is to push a new set of PSK configurations (both on fallback PSK keychain and primary PSK chain in that order) on all the association members.

The following is a sample syslog for session secured with fallback PSK:

```
%L2-MKA-5-SESSION_SECURED_WITH_FALLBACK_PSK : (Hu0/1/0/0) MKA session secured, CKN:ABCD
```

For more information on MACsec fallback PSK configuration, see Step 3, on page 12 of the Applying MACsec Configuration on an Interface, on page 12 section.

# Configuring and Verifying MACsec Encryption

MACsec can be configured on physical ethernet interfaces or member links of the interface bundles, as explained in this section.

The following section describes procedures for configuring and verifying MACsec configuration in the described deployment modes.

Prior to configuring MACsec on a router interface the MACsec keychain must be defined. If you apply the MACsec keychain on the router without specifying a MACsec policy, the default policy is applied. A default MACsec policy is pre-configured with default values. If you need to change any of the pre-configured values, create a different MACsec policy.

Configuring MACsec involves the following steps:

1. Creating a MACsec keychain
2. Creating a user-defined MACsec policy
3. Applying MACsec configuration on physical interfaces

# Creating a MACsec Keychain

A MACsec keychain is a collection of keys used to authenticate peers needing to exchange encrypted information. While creating a keychain, we define the key(s), key string with password, the cryptographic algorithm, and the key lifetime.

| MACsec Keychain Keyword | Description |
|---|---|
| Key | The MACsec key or the CKN can be up to 64 characters in length. The key must be of an even number of characters. Entering an odd number of characters will exit the MACsec configuration mode. |
| Key-string | The MACsec key-string or the CAK can be either 32 characters or 64 characters in length (32 for AES-128, 64 for AES-256). |
| Lifetime | This field specifies the validity period of a key. It includes a start time, and an expiry time. We recommend you to set the value for expiry time as *infinite*. |

**Guidelines for Configuring MACsec Keychain**

MACsec keychain management has the following configuration guidelines:

- To establish MKA session, ensure that the MACsec key (CKN) and key-string (CAK) match at both ends.
- MKA protocol uses the latest active key available in the Keychain. This key has the latest Start Time from the existing set of currently active keys. You can verify the values using the **show key chain** *keychain-name* command.
- Deletion or expiry of current active key brings down the MKA session resulting in traffic hit. We recommend you to configure the keys with infinite lifetime. If fallback is configured, traffic is safeguarded using fallback on expiry or deletion of primary-keychain active key.
- To achieve successful key rollover (CAK-rollover), the new key should be configured such that it is the latest active key, and kicks-in before the current key expires.
- We recommend an overlap of at least one minute for hitless CAK rollover from current key to new key.
- Start time and Expiry time can be configured with future time stamps, which allows bulk configuration for daily CAK rotation without any intervention of management agent.

## Procedure

**Step 1** Enter the global configuration mode and provide a name for the MACsec keychain; for example, mac_chain.

**Example:**

```
RP/0/RP0/CPU0:router(config)# key chain mac_chain
```

**Step 2** Enter the MACsec mode.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_chain)#macsec
```

**Step 3** Provide a name for the MACsec key.

The key can be up to 64 characters in length. The key must be of an even number of characters. Entering an odd number of characters will exit the MACsec configuration mode.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_chain-MacSec)#key 1234abcd5678
```

You can also configure a fall-back pre-shared key(PSK) to ensure that a PSK is always available to perform MACsec encryption and decryption. The fallback PSK along with the primary PSK ensures that the session remains active even if the primary PSK is mismatched or there is no active key for the primary PSK.

The configured key is the CKN that is exchanged between the peers.

**Note**

If you are configuring MACsec to interoperate with a MACsec server that is running software prior to Cisco IOS XR Release 6.1.3, then ensure that the MACsec key length is of 64 characters. You can add extra zero characters to the MACsec key so that the length of 64-characters is achieved. If the key length is lesser than 64 characters, authentication will fail.

**Step 4** Enter the key string and the cryptographic algorithm to be used for the key.

**Example:**

The key string is the CAK that is used for ICV validation by the MKA protocol.

```
! For AES 128-bit encryption

RP/0/RP0/CPU0:router(config-mac_chain-MacSec-1234abcd5678)#
key-string 12345678123456781234567812345678 cryptographic-algorithm AES-128-CMAC

! For AES 256-bit encryption

RP/0/RP0/CPU0:router(config-mac_chain-MacSec-1234abcd5678)#
key-string 1234567812345678123456781234567812345678123456781234567812345678 cryptographic
-algorithm AES-256-CMAC
```

**Note**
In this example, we have used the AES 256-bit encryption algorithm, and therefore, the key string is 64 hexadecimal characters in length. A 256-bit encryption algorithm uses a larger key that requires more rounds of hacking to be cracked. 256-bit algorithms provide better security against large mass security attacks, and include the security provided by 128-bit algorithms.

**Step 5** Enter the validity period for the MACsec key (CKN) also known as the lifetime period.

The lifetime period can be configured, with a duration in seconds, as a validity period between two dates (for example, Jan 01 2014 to Dec 31 2014), or with infinite validity.

The key is valid from the time you configure (in HH:MM:SS format). Duration is configured in seconds.

**Example:**

```
RP/0/RP0/CPU0:router(config- mac_chain-MacSec-1234abcd5678)#lifetime 05:00:00 01
January 2015 duration 1800
```

An example of configuring the lifetime for a defined period:

```
RP/0/RP0/CPU0:router(config-mac_chain-MacSec-1234abcd5678)#lifetime 05:00:00 20
february 2015 12:00:00 30 september 2015
```

An example of configuring the lifetime as infinite:

```
RP/0/RP0/CPU0:router(config-mac_chain-MacSec-1234abcd5678)#lifetime
05:00:00 01 January 2015 infinite
```

**Note**
When a key has expired, the MACsec session is torn down and running the **show macsec mka session** command does not display any information. If you run the **show macsec mka interface detail** command, the output displays ***** No Active Keys Present ***** in the PSK information.

**Step 6** Commit your configuration.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_chain-MacSec-1234abcd5678#commit
```

This completes the configuration of the MACsec keychain.

# Creating a User-Defined MACsec Policy

**Procedure**

**Step 1** Enter the global configuration mode, and enter a name (mac_policy) for the MACsec policy.

**Example:**

```
RP/0/RP0/CPU0:router# configure
RP/0/RP0/CPU0:router(config)# macsec-policy mac_policy
```

**Step 2** Configure the cipher suite to be used for MACsec encryption.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# cipher-suite GCM-AES-XPN-256
RP/0/RP0/CPU0:router(config-mac_policy)#GCM-AES-128
GCM-AES-256
GCM-AES-XPN-128
GCM-AES-XPN-256
```

**Note**
In this example, we have used the GCM-AES-XPN-256 encryption algorithm. A 256-bit encryption algorithm uses a larger key that requires more rounds of hacking to be cracked. 256-bit algorithms provide better security against large mass security attacks, and include the security provided by 128-bit algorithms. Extended Packet Numbering (XPN) is used to reduce the number of key rollovers while data is sent over high speed links. It is therefore highly recommended to use GCM-AES-XPN-256 encryption algorithm for higher data ports.

**Step 3** Configure the confidentiality offset for MACsec encryption.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# conf-offset CONF-OFFSET-30
```

**Note**
We recommend to change the offset value of the **conf-offset** <*offset_value*> command (MACsec encryption command) in only when the port is in **admin down** state (that is, when the interface is shut down). Changing the offset value otherwise may result in traffic loss.

**Step 4** Enter the key server priority.

You can enter a value between 0-255. Lower the value, higher the preference to be selected as the key server.

In this example, a value of 0 configures the router as the key server, while the other router functions as a key client. The key server generates and maintains the SAK between the two routers. The default key server priority value is 16.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# key-server-priority 0
```

**Step 5** Configure the security policy parameters, either Must-Secure or Should-Secure.

**Must-Secure**: Must-Secure imposes only MACsec encrypted traffic to flow. Hence, until MKA session is not secured, traffic will be dropped.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# security-policy must-secure
```

**Should-Secure**: Should-Secure allows unencrypted traffic to flow until MKA session is secured. After the MKA session is secured, Should-Secure policy imposes only encrypted traffic to flow.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# security-policy should-secure
```

***Table 2: MACsec Security Policies***

| MKA | | Secured MKA Session | Unsecured MKA Session |
|---|---|---|---|
| **Security Policy** | Must-secure | Encrypted traffic | Traffic drop (no Tx and no Rx) |
| | Should-secure | Encrypted traffic | Plain text or unencrypted traffic |

**Step 6** Configure data delay protection under MACsec policy.

Data delay protection allows MKA participants to ensure that the data frames protected by MACsec are not delayed by more than 2 seconds. Each SecY uses MKA to communicate the lowest PN used for transmission with the SAK within two seconds. Traffic delayed longer than 2 seconds are rejected by the interfaces enabled with delay protection.

By default, the data delay protection feature is disabled. Configuring the **delay-protection** command under MACsec-policy attached to MACsec interface will enable the data delay protection feature on that interface.

**Example:**

```
RP/0/RP0/CPU0:router# configure terminal
RP/0/RP0/CPU0:router(config)# macsec-policy mp1
RP/0/RP0/CPU0:router(config-macsec-policy)# delay-protection
RP/0/RP0/CPU0:router(config-macsec-policy)# commit
```

**Verification:**

The following show command output verifies that the data delay protection feature is enabled.

**Note**
Data delay protection (DDP) is not supported on the Cisco N540-24Q8L2DD-SYS routers.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec mka session interface GigabitEthernet 0/1/0/1 detail
MKA Policy Name         : mp1
Key Server Priority     : 16
Delay Protection        : TRUE
Replay Window Size      : 64
Confidentiality Offset  : 0
Algorithm Agility       : 80C201
```

```
SAK Cipher Suite        : (NONE)
MACsec Capability       : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired          : YES
```

**Step 7** Configure the replay protection window size.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# window-size 64
```

This dictates the maximum out-of-sequence frames that are accepted. You can configure a value between 0 and 1024.

**Step 8** Configure the ICV for the frame arriving on the port.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# include-icv-indicator
```

This parameter configures inclusion of the optional ICV Indicator as part of the transmitted MACsec Key Agreement PDU (MKPDU). This configuration is necessary for MACsec to interoperate with routers that run software prior to IOS XR version 6.1.3. This configuration is also important in a service provider WAN setup where MACsec interoperates with other vendor MACsec implementations that expect ICV indicator to be present in the MKPDU.

**Step 9** Commit your configuration and exit the global configuration mode.

**Example:**

```
RP/0/RP0/CPU0:router(config-mac_policy)# exit
RP/0/RP0/CPU0:router(config)# commit
RP/0/RP0/CPU0:router(config)# exit
```

**Step 10** Confirm the MACsec policy configuration.

**Example:**

```
RP/0/RP0/CPU0:router# show running-config macsec-policy

macsec-policy mac_policy
conf-offset CONF-OFFSET-30
security-policy must-secure
window-size 64
cipher-suite GCM-AES-XPN-256
key-server-priority 0
include-icv-indicator
```

This completes the configuration of the MACsec policy.

**Note**

- Small packets might be dropped when Data Delay Protection (DDP) is enabled on many MACsec enabled interfaces of a scaled setup. To avoid this, enable DDP only on the interfaces which are absolutely necessary.

# Applying MACsec Configuration on an Interface

The MACsec service configuration is applied to the host-facing interface of a CE router.

**Guidelines for MACsec Interface Configuration**

Following are the guidelines for configuring MACsec interface:

- Configure different keychains for primary and fallback PSKs.
- We do not recommend to update both primary and fallback PSKs simultaneously, because fallback PSK is intended to recover MACsec session on primary key mismatch.
- When using MACsec, we recommend you adjust the maximum transmission unit (MTU) of an interface to accommodate the MACsec overhead. Configuring MTU value on an interface allows protocols to do MTU negotiation including MACsec overhead. For instance, if the default MTU is 1514 bytes, configure the MTU to 1546 bytes (1514 + 32).
- The minimum MTU for IS-IS protocol on the MACsec interface is 1546 bytes.
- To enable MACsec on bundles:
  - Enable MACsec on all bundle members.
  - We recommend configuring the maximum possible MTU on the bundle interface.
  - The MTU configurations must account for the maximum packet size of the protocols running on the bundle interface and 32 bytes of MACsec overhead.
  - For IS-IS protocol running on the bundle interface, hello-padding must be disabled.

## Procedure

**Step 1** Enter the global configuration mode.

**Example:**

```
RP/0/RP0/CPU0:router# configure
```

**Step 2** Enter the interface configuration mode.

**Example:**

```
RP/0/RP0/CPU0:router(config)# interface Te0/3/0/1/4
```

**Step 3** Apply the MACsec configuration on an interface.

**MACsec PSK Configuration**

To apply MACsec PSK configuration on an interface, use the following command.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain mac_chain policy mac_policy
RP/0/RP0/CPU0:router(config-if)# exit
```

To apply MACsec configuration on a physical interface without the MACsec policy, use the following command.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain script_key_chain2
RP/0/RP0/CPU0:router(config-if)# exit
```

**MACsec Fallback PSK Configuration**

To apply MACsec configuration on a physical interface with a fallback PSK, use the following command.

**Example:**

```
RP/0/RP0/CPU0:router(config-if)# macsec psk-keychain mac_chain fallback-psk-keychain
fallback_mac_chain policy mac_policy
RP/0/RP0/CPU0:router(config-if)# exit
```

It is optional to configure a fallback PSK. If a fallback PSK is configured, the fallback PSK along with the primary PSK ensures that the session remains active even if the primary PSK is mismatched, or there is no active key for the primary PSK.

**Step 4** Commit your configuration.

**Example:**

```
RP/0/RP0/CPU0:router(config)# commit
```

# MACsec Policy Exceptions

By default, the MACsec security policy uses **must-secure** option, that mandates data encryption. Hence, the packets cannot be sent in clear-text format. To optionally bypass the MACsec encryption or decryption for Link Aggregation Control Protocol (LACP) packets, and to send the packets in clear-text format, use the **policy-exception lacp-in-clear** command in macsec-policy configuration mode. This functionality is beneficial in scenarios such as, in a network topology with three nodes, where bundles are terminated at the middle node, whereas MACsec is terminated at the end nodes.

This MACsec policy exception is also beneficial in interoperability scenarios where the node at the other end expects the data packets to be in clear text.

From Cisco IOS XR Software Release 7.3.1 and later, an alternative option, **allow**, is introduced under the macsec-policy configuration mode, that allows packets to be sent in clear-text format. You can use the **allow lacp-in-clear** command for LACP packets.

## How to Create MACsec Policy Exception

**Note** The **policy-exception lacp-in-clear** command under macsec-policy configuration mode is deprecated. Hence, it is recommended to use the **allow lacp-in-clear** command instead, to allow LACP packets in clear-text format.

### Configuration Example

Using the **policy-exception** command:

```
Router#configure
Router(config)#macsec-policy test-macsec-policy
Router(config-macsec-policy)#policy-exception lacp-in-clear
Router(config-macsec-policy)#commit
```

Using the **allow** command:

```
Router#configure
Router(config)#macsec-policy test-macsec-policy
Router(config-macsec-policy)#allow lacp-in-clear
Router(config-macsec-policy)#commit
```

### Running Configuration

With the **policy-exception** command:

```
Router#show run macsec-policy test-macsec-policy
macsec-policy test-macsec-policy
  policy-exception lacp-in-clear
  security-policy should-secure
  include-icv-indicator
  sak-rekey-interval seconds 120
!
```

With the **allow** command:

```
Router#show run macsec-policy test-macsec-policy
macsec-policy test-macsec-policy
  allow lacp-in-clear
  security-policy should-secure
  include-icv-indicator
  sak-rekey-interval seconds 120
!
```

### Associated Commands

- **policy-exception lacp-in-clear**
- **allow lacp-in-clear**

# Verifying MACsec Encryption on IOS XR

MACsec encryption on IOS XR can be verified by running relevant commands in the Privileged Executive Mode. The verification steps are the same for MACsec encryption on L2VPN or L3VPN network.

To verify if MACsec encryption has been correctly configured, follow these steps.

## Procedure

**Step 1** Verify the MACsec policy configuration.

**Example:**

```
RP/0/RP0/CPU0:router#show macsec policy mac_policy

==============================================================================

Policy      Cipher          Key-Svr     Window  Conf

name        Suite           Priority    Size    Offset

==============================================================================

mac_policy GCM-AES-XPN-256  0           64      30
```

If the values you see are different from the ones you configured, then check your configuration by running the **show run macsec-policy** command.

**Step 2** Verify the MACsec configuration on the respective interface.

You can verify the MACsec encryption on the configured interface bundle (MPLS network).

**Example:**

```
RP/0/RP0/CPU0:router#show macsec mka summary

NODE: node0_0_CPU0

=====================================================================
Interface     Status    Cipher Suite      KeyChain
=====================================================================

Fo0/0/0/1/0   Secured   GCM-AES-XPN-256   mac_chain

Total MACSec Sessions : 1
     Secured Sessions : 1
     Pending Sessions : 0

RP/0/RP0/CPU0:router# show macsec mka session interface Fo0/0/0/1/0
===============================================================================================
      Interface         Local-TxSCI       # Peers         Status         Key-Server
===============================================================================================
    Fo0/0/0/1/0      d46d.5023.3709/0001      1           Secured          YES
```

The **Status** field in the output confirms that the respective interface is **Secured**. If MACsec encryption is not successfully configured, you will see a status such as **Pending** or **Init**.

Run the **show run macsec-policy** command in the privileged executive mode to troubleshoot the configuration entered.

**Step 3** Verify whether the interface of the router is peering with its neighbor after MACsec configuration. The MACsec PSK validation detects inconsistency or mismatch of primary and fallback keys (CAK) being used by MKA, allowing operators to rectify the mismatch.

### Example:

The **show macsec mka session interface** *interface* **detail** command carries the Peer Validation status in the **Peer CAK** field. The values of this field can be either *Match* or *Mismatch*.

The following show command output verifies that the primary and fallback keys (CAK) are matched on both peer ends.

- ```
  RP/0/RP0/CPU0:router#show macsec mka session detail
  Peers Status:
    Last Tx MKPDU          : 2017 Sep 02 11:24:52.369
    Peer Count             : 1
    RxSCI                  : 008A960060900001
    MI                     : C2213E81C953A202C08DB999
    Peer CAK          : Match
    Latest Rx MKPDU        : 2017 Sep 02 11:24:53.360
  Fallback Data:
    CKN                    : ABCD
    MI                     : 84E724B4BA07CE414FEA84EF
    MN                     : 8
    Peers Status:
      Last Tx MKPDU        : 2017 Sep 02 11:24:52.369
      Peer Count           : 1
      RxSCI                : 008A960060900001
      MI                   : D2B902453F90389BD3385F84
      Peer CAK       : Match
      Latest Rx MKPDU      : 2017 Sep 02 11:24:53.360
  ```

- **Syslog**

  ```
  %L2-MKA-6-MKPDU_ICV_SUCCESS: (Hu0/5/0/1), ICV verification success for
  RxSCI(008a.9600.6090/0001), CKN(1000)
  %L2-MKA-6-FALLBACK_PSK_MKPDU_ICV_SUCCESS: (Hu0/5/0/1), ICV verification success for
  RxSCI(008a.9600.6090/0001), CKN(FFFF)
  ```

The following show command output verifies that the primary and fallback keys (CAK) are mismatched on both peer ends.

- ```
  RP/0/RP0/CPU0:router#show macsec mka session detail
  Peers Status:
  Last Tx MKPDU          : 2017 Sep 02 11:24:52.369
  Peer Count             : 1
  RxSCI                  : 008A960060900001
  MI                     : C2213E81C953A202C08DB999
  Peer CAK          : Mismatch
  Latest Rx MKPDU        : 2017 Sep 02 11:24:53.360
  Fallback Data:
  CKN                    : ABCD
  MI                     : 84E724B4BA07CE414FEA84EF
  MN                     : 8
  Peers Status:
    Last Tx MKPDU         : 2017 Sep 02 11:24:52.369
    Peer Count            : 1
    RxSCI                 : 008A960060900001
    MI                    : D2B902453F90389BD3385F84
    Peer CAK        : Mismatch
    Latest Rx MKPDU       : 2017 Sep 02 11:24:53.360
  ```

- **Syslog**

  ```
  %L2-MKA-3-MKPDU_ICV_FAILURE: (Hu0/5/0/1), ICV verification failed for
  RxSCI(008a.9600.6090/0001), CKN(1111)
  %L2-MKA-3-FALLBACK_PSK_MKPDU_ICV_FAILURE: (Hu0/5/0/1), ICV verification failed for
  RxSCI(008a.9600.6090/0001), CKN(9999)
  ```

The **#Peers** field in the following output confirms the presence of the peer you have configured on the physical interface, **Fo0/0/0/1/0**. If the number of peers is not reflected accurately in this output, run the **show run** command and verify the peer configuration on the interface.

```
RP/0/RP0/CPU0:router#show macsec mka session

NODE: node0_0_CPU0

============================================================
Interface    Local-TxSCI           # Peers   Status  Key-Server
============================================================

Fo0/0/0/1/0  001d.e5e9.aa39/0005      1      Secured  YES
```

**Note**
If the MKA session status is shown as **Secured** with **0 (Zero)** peer count, this means that the link is locally secured (Tx). This is because of MKA peer loss caused by **No Rx Packets (MKA Packet)** from that peer.

**Step 4** Verify whether the MKA session is secured with MACsec on the respective interface.

**Example:**

```
RP/0/RP0/CPU0:router#show macsec mka session interface Fo0/0/0/1/0 detail
MKA Detailed Status for MKA Session
==================================
Status: SECURED - Secured MKA Session with MACsec

Local Tx-SCI            : 6219.8864.e338/0001
Local Tx-SSCI           : 1
Interface MAC Address   : 6219.8864.e338
MKA Port Identifier     : 1
Interface Name          : Hu0/5/0/26
CAK Name (CKN)          : 22220000000000000000000000000000000000000000000000000000000000000000
CA Authentication Mode  : PRIMARY-PSK
Keychain                : kc
Member Identifier (MI)  : 89766F3FE9445FDCDD714CCE
Message Number (MN)     : 25711
Authenticator           : NO
Key Server              : NO
MKA Cipher Suite        : AES-256-CMAC

Latest SAK Status       : Rx & Tx
Latest SAK AN           : 0
Latest SAK KI (KN)      : E3D39135831AFBCDA7AA9DBB00000001 (1)
Old SAK Status          : FIRST-SAK
Old SAK AN              : 0
Old SAK KI (KN)         : FIRST-SAK (0)

SAK Transmit Wait Time  : 0s (Not waiting for any peers to respond)
SAK Retire Time         : 0s (No Old SAK to retire)
Time to SAK Rekey       : NA
MKA Policy Name         : *DEFAULT POLICY*
Key Server Priority     : 16
Replay Window Size      : 64
Confidentiality Offset  : 0
Algorithm Agility       : 80C201
SAK Cipher Suite        : 0080C20001000004 (GCM-AES-XPN-256)
MACsec Capability       : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired          : YES

# of MACsec Capable Live Peers           : 1
# of MACsec Capable Live Peers Responded : 0
```

```
Live Peer List:
  MI                        MN          Rx-SCI (Peer)        SSCI KS-Priority
  ------------------------------------------------------------------------------

RP/0/RP0/CPU0:router#show macsec mka session interface Fo0/0/0/1/0 detail Tue May 18
13:23:29.935 UTC
Tue May 18 13:23:29.935 UTC

MKA Detailed Status for MKA Session
===================================
Status: Secured - Secured MKA Session with MACsec

Local Tx-SCI                    : 008a.96d6.194c/0001
Local Tx-SSCI                   : 2
Interface MAC Address           : 008a.96d6.194c
MKA Port Identifier             : 1
Interface Name                  : Hu0/2/0/11
CAK Name (CKN)                  : 2111
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : test1
Member Identifier (MI)          : 69B39E87B3CBA673401E9891
Message Number (MN)             : 352
Authenticator                   : NO
Key Server                      : YES
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128
Key Distribution Mode           : SAK

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 69B39E87B3CBA673401E989100000001 (1)
Old SAK Status                  : FIRST-SAK
Old SAK AN                      : 0
Old SAK KI (KN)                 : FIRST-SAK (0)

SAK Transmit Wait Time          : 0s (Not waiting for any peers to respond)
SAK Retire Time                 : 0s (No Old SAK to retire)
Time to SAK Rekey               : 456s
Time to exit suspension         : NA

MKA Policy Name                 : P12
Key Server Priority             : 20
Delay Protection                : TRUE
Replay Window Size              : 100
Include ICV Indicator           : TRUE
Confidentiality Offset          : 0
Algorithm Agility               : 80C201
SAK Cipher Suite                : 0080C20001000003 (GCM-AES-XPN-128)
MACsec Capability               : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired                  : YES

# of MACsec Capable Live Peers          : 1
# of MACsec Capable Live Peers Responded : 1

# of MACSec Suspended Peers             : 0

Live Peer List:
------------------------------------------------------------------------------
           MI                  MN             Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------
42A78BD6243539E917B8C6B2      290       7061.7bea.1df4/0001    1        20

Potential Peer List:
------------------------------------------------------------------------------
```

```
          MI                  MN            Rx-SCI          SSCI  KS-Priority
------------------------------------------------------------------------------

Suspended Peer List:
------------------------------------------------------------------------------
       Rx-SCI          SSCI
------------------------------------------------------------------------------

Peers Status:
Last Tx MKPDU          : 2021 May 18 13:23:29.588
Peer Count             : 1

 RxSCI                  : 70617BEA1DF40001
  MI                    : 42A78BD6243539E917B8C6B2
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 May 18 13:23:29.847

MKA Detailed Status for MKA Session
===================================
Status: Active - Marked Peer as Live (Waiting for SAK generation/distribution)

Local Tx-SCI                    : 008a.96d6.194c/0001
Local Tx-SSCI                   : 2
Interface MAC Address           : 008a.96d6.194c
MKA Port Identifier             : 1
Interface Name                  : Hu0/2/0/11
CAK Name (CKN)                  : 2000
CA Authentication Mode          : FALLBACK-PSK
Keychain                        : test1f
Member Identifier (MI)          : 8F59AD6021FA3E2D5F9E6231
Message Number (MN)             : 350
Authenticator                   : NO
Key Server                      : YES
MKA Cipher Suite                : AES-128-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-128
Key Distribution Mode           : SAK

Latest SAK Status               : Rx & Tx
Latest SAK AN                   : 0
Latest SAK KI (KN)              : 69B39E87B3CBA673401E989100000001 (1)
Old SAK Status                  : FIRST-SAK
Old SAK AN                      : 0
Old SAK KI (KN)                 : FIRST-SAK (0)

SAK Transmit Wait Time          : 0s (Not waiting for any peers to respond)
SAK Retire Time                 : 0s (No Old SAK to retire)
Time to SAK Rekey               : 456s
Time to exit suspension         : NA

MKA Policy Name                 : P12
Key Server Priority             : 20
Delay Protection                : TRUE
Replay Window Size              : 100
Include ICV Indicator           : TRUE
Confidentiality Offset          : 0
Algorithm Agility               : 80C201
SAK Cipher Suite                : 0080C20001000003 (GCM-AES-XPN-128)
MACsec Capability               : 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired                  : YES

# of MACsec Capable Live Peers           : 1
# of MACsec Capable Live Peers Responded : 0

# of MACSec Suspended Peers              : 0
```

```
Live Peer List:
----------------------------------------------------------------------------------
           MI                   MN             Rx-SCI          SSCI  KS-Priority
----------------------------------------------------------------------------------
1BB9428C721F6EE3E538C942      288       7061.7bea.1df4/0001    1        20

Potential Peer List:
----------------------------------------------------------------------------------
           MI                   MN             Rx-SCI          SSCI  KS-Priority
----------------------------------------------------------------------------------

Suspended Peer List:
----------------------------------------------------------------------------------
       Rx-SCI          SSCI
----------------------------------------------------------------------------------

Peers Status:
Last Tx MKPDU          : 2021 May 18 13:23:29.587
Peer Count             : 1

RxSCI                  : 70617BEA1DF40001
  MI                    : 1BB9428C721F6EE3E538C942
  Peer CAK              : Match
  Latest Rx MKPDU       : 2021 May 18 13:23:29.847

RP/0/RP0/CPU0:router#
```

The **Status** field in the output verifies if the MKA session is secured with MACsec encryption. The output also displays information about the interface and other MACsec parameters.

**Step 5** Verify the MACsec session counter statistics.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec mka statistics interface Fo0/0/0/1/0


MKA Statistics for Session on interface (Fo0/0/0/1/0)
=====================================================
Reauthentication Attempts.. 0

CA Statistics
Pairwise CAKs Derived... 0
Pairwise CAK Rekeys..... 0
Group CAKs Generated.... 0
Group CAKs Received..... 0

SA Statistics
SAKs Generated.......... 3
SAKs Rekeyed............ 2
SAKs Received........... 0
SAK Responses Received.. 3

MKPDU Statistics
MKPDUs Transmitted...... 5425
"Distributed SAK".. 8
"Distributed CAK".. 0
MKPDUs Validated & Rx... 4932
"Distributed SAK".. 0
"Distributed CAK".. 0

MKA IDB Statistics
```

```
MKPDUs Tx Success.......... 5425
MKPDUs Tx Fail............. 0
MKPDUS Tx Pkt build fail... 0
MKPDUs Rx CA Not found..... 0
MKPDUs Rx Error............ 0
MKPDUs Rx Success.......... 4932

MKPDU Failures
   MKPDU Rx Validation (ICV)............... 0
   MKPDU Rx Bad Peer MN.................... 0
   MKPDU Rx Non-recent Peerlist MN........ 0
   MKPDU Rx Drop SAKUSE, KN mismatch...... 0
   MKPDU Rx Drop SAKUSE, Rx Not Set....... 0
   MKPDU Rx Drop SAKUSE, Key MI mismatch.. 0
   MKPDU Rx Drop SAKUSE, AN Not in Use.... 0
   MKPDU Rx Drop SAKUSE, KS Rx/Tx Not Set. 0

SAK Failures
   SAK Generation................... 0
   Hash Key Generation.............. 0
   SAK Encryption/Wrap.............. 0
   SAK Decryption/Unwrap............ 0
```

The counters display the MACsec PDUs transmitted, validated, and received. The output also displays transmission errors, if any.

This completes the verification of MACsec encryption on the IOS-XR.

# Verifying MACsec Encryption on the Router

MACsec encryption on the router hardware can be verified by running relevant commands in the Privileged Executive Mode.

To verify if MACsec encryption has been correctly configured, follow these steps.

### Procedure

**Step 1** Verify the MACsec encryption and hardware interface descriptor block (IDB) information on the interface.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec ea idb interface Fo0/0/0/1/0


IDB Details:
if_sname : Fo0/0/0/1/0
if_handle : 0x3480
Replay window size : 64
Local MAC : 00:1d:e5:e9:aa:39
Rx SC Option(s) : Validate-Frames Replay-Protect
Tx SC Option(s) : Protect-Frames Always-Include-SCI
Security Policy : MUST SECURE
Sectag offset : 8
Rx SC 1
Rx SCI : 001de5e9b1bf0019
```

```
Peer MAC : 00:1d:e5:e9:b1:bf
Stale : NO
SAK Data
SAK[0] : ***
SAK Len : 32
HashKey[0] : ***
HashKey Len : 16
Conf offset : 30
Cipher Suite : GCM-AES-XPN-256
CtxSalt[0] : 83 c3 7b ad 7b 6f 63 16 09 8f f3 d2
Rx SA Program Req[0]: 2015 Oct 09 15:20:53.082
Rx SA Program Rsp[0]: 2015 Oct 09 15:20:53.092

Tx SC
Tx SCI : 001de5e9aa39001a
Active AN : 0
Old AN : 255
Next PN : 1, 0, 0, 0
SAK Data
SAK[0] : ***
SAK Len : 32
HashKey[0] : ***
HashKey Len : 16
Conf offset : 30
Cipher Suite : GCM-AES-XPN-256
CtxSalt[0] : 83 c3 7b ae 7b 6f 63 16 09 8f f3 d2
Tx SA Program Req[0]: 2015 Oct 09 15:20:55.053
Tx SA Program Rsp[0]: 2015 Oct 09 15:20:55.064
```

The **if_handle** field provides the IDB instance location.

The **Replay window size** field displays the configured window size.

The **Security Policy** field displays the configured security policy.

The **Local Mac** field displays the MAC address of the router.

The **Peer Mac** field displays the MAC address of the peer. This confirms that a peer relationship has been formed between the two routers.

**Step 2** Use the IDB handle retrieved from Step 1 to verify the platform hardware information.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec platform hardware
idb location 0/0/CPU0 | b 3480

if_handle : 0x00003480
NPPort : 099 [0x063]
LdaPort : 016 [0x010] SerdesPort : 000 [0x000]
NetSoftPort : 061 [0x03d] SysSoftPort : 062 [0x03e]
Active AN : 0x00000000 Idle AN : 0x000000ff
Match-All Tx SA : 0x80010001 Match-All Rx SA : 0x00010001
Match-All Tx Flow : 0x80000003 Match-All Rx Flow : 0x00000003
Bypass Tx SA : 0x80000000 Bypass Rx SA : 0x00000000
Tx SA[0] : 0x80020002 Tx Flow[0] : 0x8000000c
Tx SA[1] : 0xffffffff Tx Flow[1] : 0xffffffff
Tx SA[2] : 0xffffffff Tx Flow[2] : 0xffffffff
Tx SA[3] : 0xffffffff Tx Flow[3] : 0xffffffff
Rx SA[0] : 0x00020002 Rx Flow[0] : 0x0000000c
Rx SA[1] : 0xffffffff Rx Flow[1] : 0xffffffff
Rx SA[2] : 0xffffffff Rx Flow[2] : 0xffffffff
```

```
Rx SA[3] : 0xffffffff Rx Flow[3] : 0xffffffff
```

**Step 3** Use the Transmitter SA retrieved from Step 2 to verify the MACsec SA information programmed in the hardware.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec platform hardware sa
0x80020002 interface Fo0/0/0/1/0 location 0/0/CPU0

MACsec HW SA Details:
Action Type : 0x00000003
Direction : Egress
Dest Port : 0x00000000
Conf Offset : 00000030
Drop Type : 0x00000002
Drop NonResvd : 0x00000000
SA In Use : YES
ConfProtect : YES
IncludeSCI : YES
ProtectFrame : YES
UseEs : NO
UseSCB : NO
SCI : 00 1d e5 e9 aa 39 00 05
Replay Window : 64 MacsecCryptoAlgo : 7
Direction : Egress AN : 0
AES Key Len : 256 X-Packet Number : 0x0000000000000000
CtxSalt : f8d88dc3e1c5e6a94ca2299
```

The output displays the details of the encryption, such as the AES key, the Auth key, and other parameters.

**Step 4** Verify the MACsec Secure Channel (SC) information programmed in the hardware.

**Example:**

```
RP/0/RP0/CPU0:router# show macsec platform hardware msc
interface Fo0/0/0/1/0 location 0/0/CPU0


MACsec HW Cfg Details:
Mode : 0x5
Counter Clear on Read : 0x0
SA Fail Mask : 0xffff
Global SecFail Mask : 0xffffffff
Latency : 0xff
StaticBypass : 0x0
Should secure : 0x0
Global Frame Validation : 0x2
Ctrl Pkt CC Bypass : 0x1
NonCtrl Pkt CC Bypass : 0x1
Sequence Number Threshold : 0xbffffb8
Sequence Number Threshold 64bit : 0x000002fffffffffd
Non Matching Non Control Pkts Programming
      Untagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
      Tagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
      BadTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
      KayTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
Non Matching Control Pkts Programming
      Untagged : Bypass: 0x1 DestPort : 0x2, DropType : 0xffffffff
```

```
Tagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
BadTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
KayTagged : Bypass: 0x0 DestPort : 0x2, DropType : 0x2
```

This completes the verification of MACsec encryption on the router hardware.

This completes the configuration and verification of MACsec encryption.

# Quantum safe key distribution options for MACsec

Quantum computers are a threat to existing cryptographic algorithms. To address this problem, you can use session keys to establish a secure connection between two routers.

Cisco offers two solutions to derive session keys:

- Session Key Service (SKS): Used to derive the session keys on both the routers establishing the MACsec connection without using an external key source.
- Secure Key Integration Protocol (SKIP): Used to derive the session keys on both the routers establishing the MACsec connection using an external server. Enables a router to securely import a post-quantum pre-shared key (PPK) from an external key source such as a quantum key distribution (QKD) device.

***Table 3: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Session key service | Release 7.10.1 | The Session Key Service (SKS) is now supported on:<br>• N540-ACC-SYS<br>• N540X-ACC-SYS<br>• N540-24Z8Q2C-SYS |
| Session key service | Release 7.9.1 | The router integrates the Session Key Service (SKS) as a software component, allowing it to generate and manage the cryptographic keys needed for quantum-safe MACsec. By using SKS, you can implement MACsec without requiring additional hardware, simplifying deployment and reducing costs. The SKS software should be present on the peer routers.<br>Unsupported platforms:<br>• N540-ACC-SYS<br>• N540X-ACC-SYS<br>• N540-24Z8Q2C-SYS<br>For more information on Quantum Key Distribution, see Post Quantum Security Brief. |

# Session key service

The Session Key Service (SKS) is a cryptographic service that manages symmetric keys for encryption and decryption. These are the salient features of SKS on a router.

**Key generation**

The SKS engine is a software component within the router responsible for generating cryptographic keys. It creates keys that will be used to encrypt and decrypt data between peer routers.

**No additional hardware required**

This implies that the key generation and exchange process does not need extra hardware components. The SKS engine functions with the existing router hardware, making it cost-effective and easy to deploy.

**Seed protected by McEliece cryptosystem**

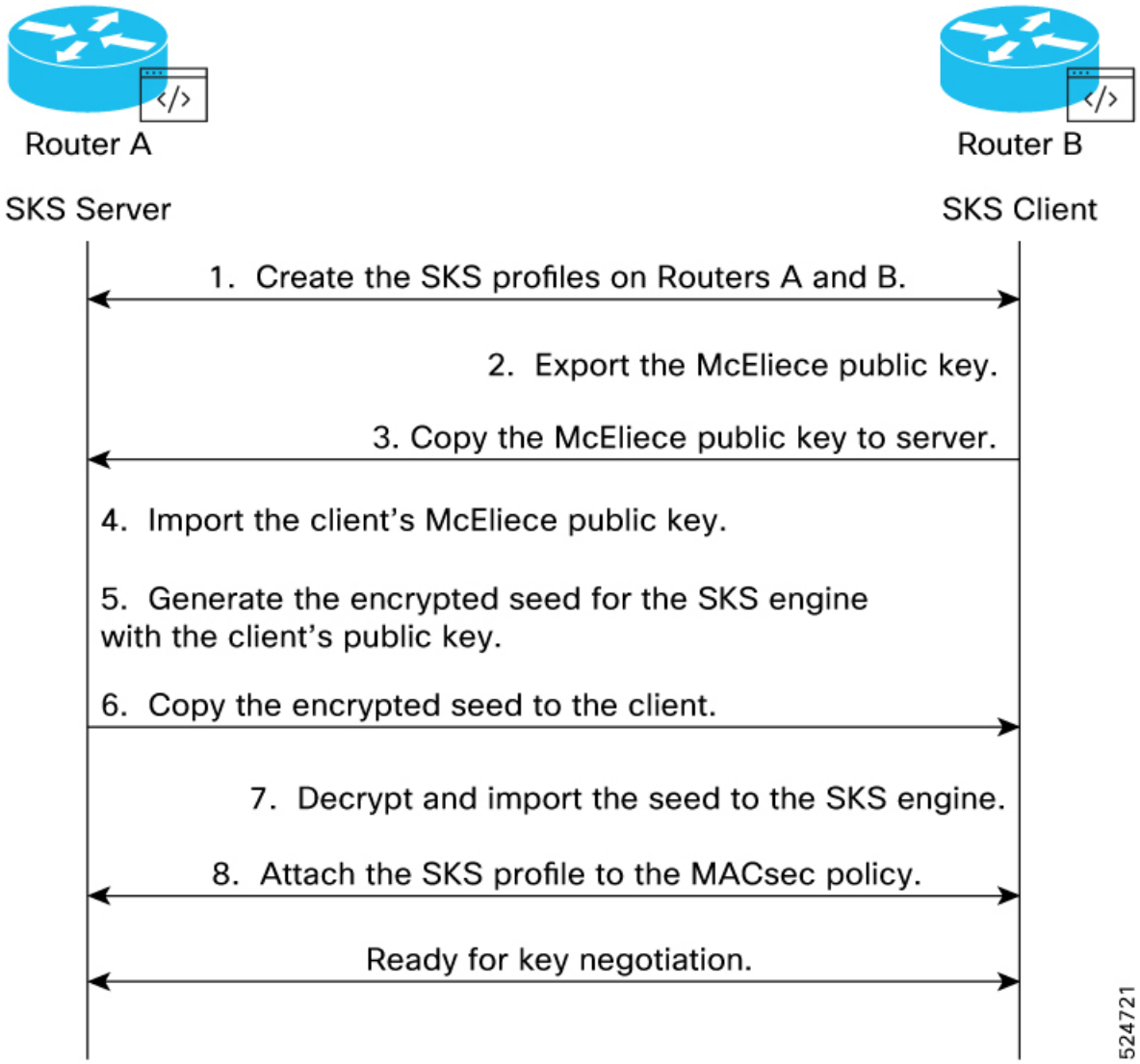
Seeding refers to initializing the SKS with a specific value, called a seed, which ensures that both communicating peers generate the same cryptographic keys. This is crucial for successful encryption and decryption. The McEliece cryptosystem is a public-key cryptosystem known for its resistance to quantum computer attacks. Protecting the seed with McEliece ensures it remains secure against future quantum computing threats.

**Only Key ID sent on the network**

Instead of sending the entire cryptographic key over the network, the router sends only a Key Identifier (Key ID). The receiving peer uses the Key ID to derive the corresponding key locally using its SKS. Using the Key ID enhances security by minimizing exposure of the actual key during transmission.

# Configure SKS

The SKS software component on a router is used to configure the server and client to generate preshared keys for quantum-safe MACsec. This image depicts the steps you should perform to generate preshared keys. You can click a step to learn more.




**1** Create the SKS profiles on Routers A and B.
**2** Export the McEliece public key.
**3** Copy the McEliece public key to the server.
**4** Import the client's McEliece public key.
**5** Generate the encrypted seed for the SKS engine with the clients public key.
**6** Copy the encrypted seed to the client.
**7** Decrypt and import the seed to the sks engine.
**8** Attach the SKS profile to the MACsec policy.
**9** Ready for key negotiation.

## Create the SKS profile on the server and client

Follow these steps to create the SKS profile on the server and client.

### Before you begin

First, let us gather the required details to facilitate the devices to agree on encryption parameters to establish a secure connection:

- Routers A and B are equipped with the SKS engine.
- Router A acts as the server and Router B acts as the client.
- Routers A and B are ready for key negotiation after all the steps are performed and the SKS profile is attached to the MACsec policy.

### Procedure

**Step 1** Enter the **sks profile** <*profile-name*> **device-identifier** <*name of peer*> command to apply a profile to manage secure communications.

**Server configuration**

```
Router A(config)# sks profile prof-A device-identifier peer-1
```

**Client configuration**

```
Router B(config)#sks profile prof-B device-identifier peer-2
```

**Step 2** Enter the **live-key** <*number of MACsec sessions*> command to manage the active keys used in MACsec sessions.

**Server configuration**

```
Router A(config-sks-profile)#live-keys 5
```

**Client configuration**

```
Router B(config-sks-profile)#live-keys 5
```

**Step 3** Enter the **peer identifier** <*peer-name*> command to associate the server with the client and the client with the server.

**Server configuration**

```
Router A(config-sks-profile)#peer-identifier peer-2
```

**Client configuration**

```
Router B(config-sks-profile)#peer-identifier peer-1 master
```

## Export the McEliece public key

Follow these steps to export the McEliece public key.

### Procedure

**Step 1** Enter the **crypto sks key export mceliece** command to export the public key to the client.

**Client configuration**

```
Router B#crypto sks key export mceliece
```

The message **Pubkey exported file: disk0:/MeCe_the_MC_default_pub** is displayed.

**Step 2** Verify the default path where the public key is exported.

**Client configuration**

```
Router B#dir disk0:/MeCe_the_MC_default_pub
Mon Feb 24 04:25:25.013 UTC

Directory of disk0:/MeCe_the_MC_default_pub
73 -rw-r--r--. 1 1357824 Feb 24 04:20 MeCe_the_MC_default_pub

989244 kbytes total (919872 kbytes free)
```

## Copy the McEliece public key to the server

Follow these steps to copy the McEliece public key to the server.

### Before you begin

In the example, **disk0:/MeCe_the_MC_default_pub** is the source path of the client and **cisco@1.2.42.3** is the IP address of the server.

### Procedure

**Step 1** Using the Secure Copy Protocol (SCP), enter the **scp / disk0\:/***<source path on the client <destination path of the server***:/disk0:/** to copy the files from the client to the server.

**Client configuration**

```
Router B# scp /disk0\:/MeCe_the_MC_default_pub cisco@1.2.42.3:/disk0:/
```

This command has securely copied the file named **MeCe_the_MC_default_pub** from the local directory **/disk0:/** to the remote directory **/disk0:/** on the host **1.2.42.3** using the Cisco user account.

**Step 2** Verify that the files are copied from the client to the server.

**Client configuration**

```
Router B #dir disk0:/MeCe_the_MC_default_pub
Mon Feb 24 04:27:00.398 UTC

Directory of disk0:/MeCe_the_MC_default_pub
73 -rw-r--r--. 1 1357824 Feb 24 04:26 MeCe_the_MC_default_pub

989244 kbytes total (919868 kbytes free)
```

## Import the client McEliece public key

Follow these steps to import the client McEliece public key.

### Before you begin

In the example, peer 2 is Router B's name to which the key corresponds. The local directory **disk0:/MeCe_the_MC_default_pub** is the source path of the key on the server that was copied in the previous step.

### Procedure

**Step 1** Enter the **crypto sks key import mceliece** <*client's name*> <*source path of the key on the server*> command to import the McEliece public key to the server.

**Server configuration**

```
Router A# crypto sks key import mceliece peer-2 disk0:/MeCe_the_MC_default_pub
```

**Step 2** Verify that the **Pubkey Import Done** is set to **True** for the required peer.

**Server configuration**

```
Router A# show crypto sks peer all
Mon Feb 24 04:29:06.492 UTC
Peer Name          : peer-2
Profile Name       : prof-A
Seed Done          : TRUE
Pubkey Import Done : TRUE
Master             : FALSE
```

## Generate the encrypted seed for the SKS engine with the client public key

Follow these steps to generate the encrypted seed for the SKS engine with the client public key.

### Procedure

**Step 1** Enter the **crypto sks seed export mceliece** <*client name*> command to generate and export the seed to the server.

**Server configuration**

```
Router A# crypto sks seed export mceliece peer-2
```

The command exports a McEliece cryptographic seed that is associated with Router B.

**Step 2** Verify if an encrypted seed is exported to the */disk0\:/enc_self_peer-2* location.

**Server configuration**

```
Router A# dir disk0:/enc_self_peer-2
Mon Feb 24 04:35:57.596 UTC

Directory of disk0:/enc_self_peer-2
74 -rw-r--r--. 1 480 Feb 24 04:35 enc_self_peer-2

989244 kbytes total (919860 kbytes free)
```

### Copy the encrypted seed to the client

Follow these steps to copy the encrypted seed to the client.

### Procedure

**Step 1** Using the Secure Copy Protocol (SCP), enter the **scp /disk0\:/**<*source path of the client*> <*destination path of the server*> command to copy the files from the server to the client.

**Server configuration**

```
Router A# scp /disk0\:/enc_self_peer-2 cisco@1.2.43.3:/disk0:/
```

**Step 2** Verify that the files are copied from the server to the client.

**Client configuration**

```
Router B# dir disk0:/enc_self_peer-2
Mon Feb 24 04:35:57.596 UTC

Directory of disk0:/enc_self_peer-2
74 -rw-r--r--. 1 480 Feb 24 04:35 enc_self_peer-2

989244 kbytes total (919860 kbytes free)
```

## Decrypt and import the seed to the sks engine

Follow these steps to decrypt and import the seed to the sks engine.

### Procedure

**Step 1** Enter the **crypto sks seed import mceliece** <*server name*> <*server path*> command to import the seed on the client.

**Client configuration**

```
Router B# crypto sks seed import mceliece peer-1 disk0:/enc_self_peer-2
```

The seed is associated with Router A and **disk0:/enc_self_peer-2** is the file path from which the seed is being imported. It indicates that the seed is stored in a file located at **disk0:/enc_self_peer-2** on the router.

**Step 2** Verify that the seed is imported to the client.

**Client configuration**

```
Router B# show crypto sks peer all
Mon Feb 24 04:37:35.578 UTC
Peer Name          : peer-1
Profile Name       : prof-B
Seed Done          : TRUE
Pubkey Import Done : FALSE
Master             : TRUE
------------------------------
```

## Attach the SKS profile to the MACsec policy

Follow these steps to create the SKS profile on the server and client.

### Procedure

**Step 1** Configure the SKS profile on both MACsec peers.

**Server configuration**

```
Router A(config)#macsec-policy p1
Router A(config-macsec-policy-p1)#ppk
Router A(config-macsec-policy-p1-ppk)#sks-profile prof-A
```

**Client configuration**

```
Router B(config)#macsec-policy p2
Router B(config-macsec-policy-p1)#ppk
Router B(config-macsec-policy-p1-ppk)#sks-profile prof-B
```

**Step 2** Verify the MACsec configuration.

**Server configuration**

```
Router A#show macsec mka session

MKA Detailed Status for MKA Session
===================================
Status: Secured - Secured MKA Session with MACsec

Local Tx-SCI                    : c847.091c.d060/0001
Local Tx-SSCI                   : 1
Interface MAC Address           : c847.091c.d060
MKA Port Identifier             : 1
Interface Name                  : Te0/0/0/24
CAK Name (CKN)                  : 4000
CA Authentication Mode          : PRIMARY-PSK
Keychain                        : tet
Member Identifier (MI)          : 2E222A17F6E9535E1ACD4747
Message Number (MN)             : 333807
Authenticator                   : NO
Key Server                      : NO
MKA Cipher Suite                : AES-256-CMAC
Configured MACSec Cipher Suite  : GCM-AES-XPN-256
Key Distribution Mode           : PPK
------------<truncated>--------------
```

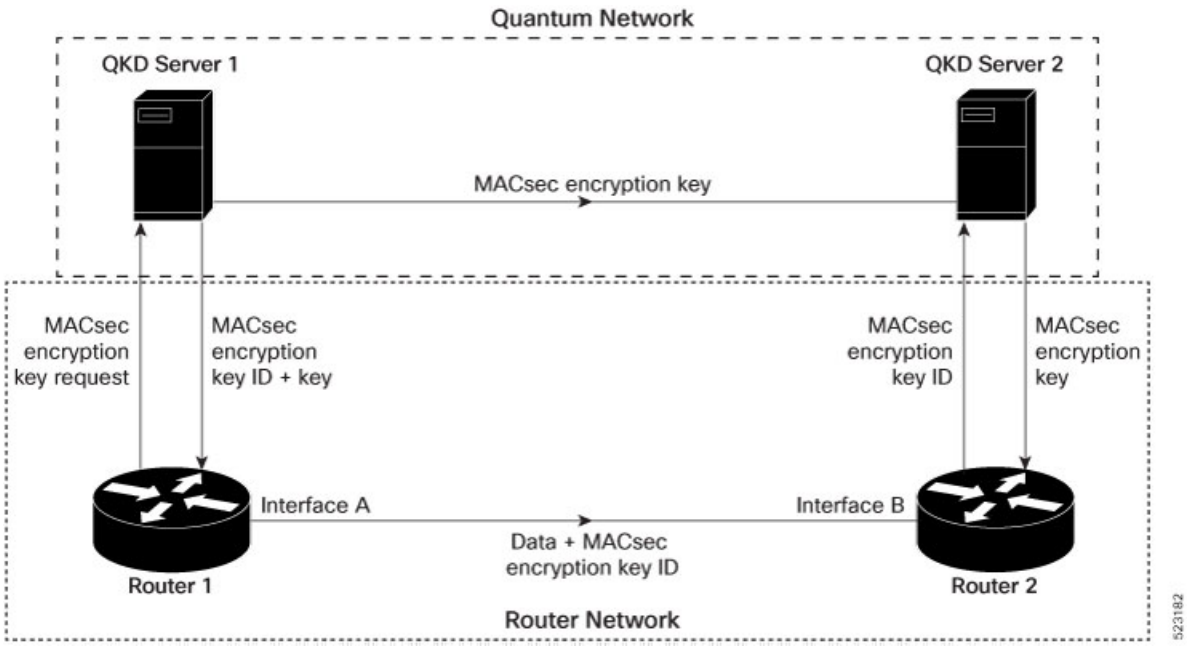
**Ready for key negotiation**: Once the routers attach the SKS profile to the MACsec policy, they have met all technical and security prerequisites. Routers A and B can now establish a secure communication link using symmetric key cryptography.

# Secure Key Integration Protocol

*Table 4: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Secure Key Integration Protocol for Routers | Release 7.9.1 | Your routers are now capable of handling the Secure Key Integration Protocol (SKIP) protocol. The SKIP protocol enables your routers to communicate with external quantum devices. With this ability, you can use the Quantum Key Distribution (QKD) devices for exchanging MACsec encryption keys between routers. Using QKD eliminates the key distribution problem in a post quantum world where the current cryptographic systems are no longer secure due to the advent of quantum computers.<br>This feature introduces the following:<br>• **CLI:**<br>  • **crypto-sks-kme**<br>  • **show crypto sks profile**<br>• **Yang Data Model**: Cisco-IOS-XR-um-sks-server-cfg.yang (see GitHub, YANG Data Models Navigator)<br>For more information on Quantum Key Distribution, see Post Quantum Security Brief. |

Cisco Secure Key Integration Protocol (SKIP) enables your router that supports encryption to use keys by a quantum distribution system. SKIP implementation in Cisco IOS-XR software supports integrating external Quantum Key Distribution (QKD) devices with your routers. With integration support between the routers and QKD devices, you can use the QKD devices to exchange encryption keys for communication between the routers. And this mechanism eliminates the key distribution problem in a post quantum world.




Quantum Key Distribution (QKD) is a method for securely transmitting a secret key between two parties. QKD uses the laws of quantum mechanics to guarantee security even when eavesdroppers monitor the communication channel. In QKD, the key is encoded in the states of single photons. The QKD transmits the keys over optical fiber or free space (vacuum). The security of the key relies on the fact that measuring a quantum state introduces a change in the quantum state. The change in quantum states helps the two end parties of the communication channel to identify any interception of their key.

QKD is a secure key exchange mechanism against quantum attacks and will remain so, even with future advancements in cryptanalysis or quantum computing. Unlike other cryptographic algorithms, QKD doesn't need continual updates based on discovered vulnerabilities.

### Feature Highlights

- You can use the QKD devices in the following combinations:
  - Same QKD device on the end ports of the peer routers
  - Different QKD devices on the end ports of the peer routers
  - Multiple links between the same peer routers using different QKD devices
- You can use a specific source interface for the router communication with the QKD devices. To use a specific source interface, configure the source interface in the QKD profile. Use the **source interface** command in SKS configuration mode as follows.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# source interface hundredGigE 0/1/0/17
Router(config-sks-profile)# commit
```

- You can use an HTTP Proxy for the router communication with the QKD devices. Use the following configuration for the router to use an HTTP proxy server to communicate to the QKD devices.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# http proxy ipv4 192.0.2.68 port 804
Router(config-sks-profile)# commit
```

**Note** The **http proxy server** command supports configuration using IPv4 address, IPv6 address, and hostname of the HTTP proxy.

### Restrictions

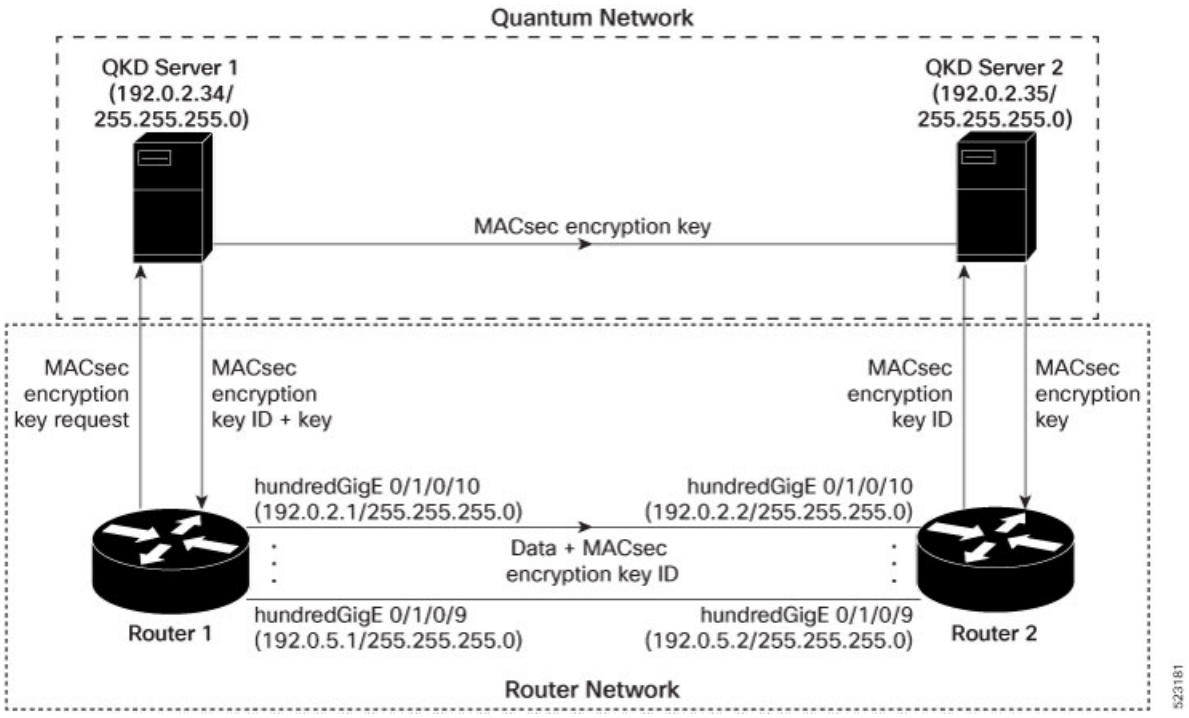
The following section lists the restriction to consider prior to implementing SKIP:

- You can use the SKIP protocol only in a Point to Point MACsec link encryption scenario.
- The SKIP protocol is available only on the interfaces that support MACsec encryption.

# Configuring Point to Point MACsec Link Encryption using SKIP

In Point-to-Point MACsec Link Encryption, the router uses SKIP to establish secure encryption. This encryption is set up between two interfaces in peer routers and requires the assistance of an external QKD device network. The QKD network shares the MACsec encryption key instead of the router network. Thus, when the router needs to create a MACsec link between peer router interfaces, it contacts the external QKD device and requests the key. The external QKD device generates a Key pair comprising the Key ID and the Key. The Key ID serves as the unique identification string for the Key (Shared Secret). The QKD then shares both the Key ID and Key with the router and the router shares only the Key ID with its peer. The Peer router uses this Key ID to retrieve encryption keys from its QKD device. Therefore, Quantum networks securely communicate encryption keys always.

*Figure 3: Point to Point MACsec Link Encryption using SKIP*




## Prerequisites

- Configure MACsec Pre-Sared Key (PSK). For more information, see MACsec PSK, on page 5.
- Configure MACsec in the PPK mode.
- An external QKD devices network.
- Add the QKD server CA to the trustpoint in the router. For more information, see Configure Trustpoint.
- Import the QKD server root CA certificate in the router. For more information, see Configure Certificate Enrollment Using Cut-and-Paste.

## Configuration

The following example details how to establish Point to Point MACsec Link Encryption using SKIP:

**Router 1:**

1. Configure the QKD profile.

```
Router# config
Router(config)# sks profile ProfileR1toR2 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.34 port 10001
Router(config-sks-profile)# commit
```

2. Map the QKD profile to the MACsec policy.

```
Router# config
Router(config)# macsec-policy R1toR2
Router(config-macsec-policy)# ppk sks-profile ProfileR1toR2
Router(config-macsec-policy)# commit
```

**Note** For more information on MACsec Policy, see Creating a User-Defined MACsec Policy, on page 9.

3. Apply MACsec policy to the interfaces.

```
Router# config
Router(config)#interface hundredGigE 0/1/0/10
Router(config-if)# ipv4 address 192.0.2.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/11
Router(config-if)# ipv4 address 192.0.3.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/12
Router(config-if)# ipv4 address 192.0.4.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
Router(config)#interface hundredGigE 0/1/0/9
Router(config-if)# ipv4 address 192.0.5.1 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R1toR2
Router(config)# commit
```

**Router 2:**

1. Configure the QKD profile.

```
Router# config
Router(config)# sks profile ProfileR2toR1 type remote
Router(config-sks-profile)# kme server ipv4 192.0.2.35 port 10001
Router(config-sks-profile)# commit
```

2. Map the QKD profile to the MACsec policy.

```
Router# config
Router(config)# macsec-policy R2toR1
Router(config-macsec-policy)# ppk sks-profile ProfileR2toR1
Router(config-macsec-policy)# commit
```

**Note** For more information on MACsec Policy, see Creating a User-Defined MACsec Policy, on page 9.

3. Apply MACsec policy to the interfaces.

```
Router# config
Router(config)#interface hundredGigE 0/1/0/10
Router(config-if)# ipv4 address 192.0.2.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
Router(config)#interface hundredGigE 0/1/0/11
Router(config-if)# ipv4 address 192.0.3.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
Router(config)#interface hundredGigE 0/1/0/12
Router(config-if)# ipv4 address 192.0.4.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
Router(config-if)# commit
Router(config)#interface hundredGigE 0/1/0/9
Router(config-if)# ipv4 address 192.0.5.2 255.255.255.0
Router(config-if)# macsec psk-keychain mac_chain policy R2toR1
```

```
Router(config-if)# commit
```

### Running Configuration

#### Router 1:

```
sks profile ProfileR1toR2 type remote
 kme server ipv4 192.0.2.34 port 10001
!
macsec-policy R1toR2
 ppk
  sks-profile ProfileR1toR2
 !
!
interface hundredGigE 0/1/0/10
 ipv4 address 192.0.2.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/11
 ipv4 address 192.0.3.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/12
 ipv4 address 192.0.4.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
interface hundredGigE 0/1/0/9
 ipv4 address 192.0.5.1 255.255.255.0
 macsec psk-keychain mac_chain policy R1toR2
!
```

#### Router 2:

```
sks profile ProfileR2toR1 type remote
 kme server ipv4 192.0.2.35 port 10001
!
macsec-policy R2toR1
 ppk
  sks-profile ProfileR2toR1
 !
!
interface hundredGigE 0/1/0/10
 ipv4 address 192.0.2.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!t
interface hundredGigE 0/1/0/11
 ipv4 address 192.0.3.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
interface hundredGigE 0/1/0/12
 ipv4 address 192.0.4.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
interface hundredGigE 0/1/0/9
 ipv4 address 192.0.5.2 255.255.255.0
 macsec psk-keychain mac_chain policy R2toR1
!
```

## Verification

```
Router(config)# show crypto sks profile all
Profile Name        :ProfileR1toR2
Myidentifier       :Router1
Type               :Remote
Reg Client Count     :1

Server
IP                  :192.0.2.34
Port                :10001
Vrf                 :Notconfigured
Source Interface    :Notconfigured
Status              :Connected
Entropy             :true
Key                 :true
Algorithm           :QKD
Local identifier    :Alice
Remote identifier   :Alice

Peerlist
QKD ID               :Bob
State                :Connected

Peerlist
QKD ID               :Alice
State                :Connected


Router# show crypto sks profile all stats
Profile Name            : ProfileR1toR2
My identifier           : Router1
Server
 IP                     : 192.0.2.34
 Port                   : 10001
 Status                 : connected
Counters
 Capability request         : 1
 Key request                : 3
 Key-id request             : 0
 Entropy request            : 0
 Capability response        : 1
 Key response               : 3
 Key-id response            : 0
 Entropy response           : 0
 Total request              : 4
 Request failed             : 0
 Request success            : 4
 Total response             : 4
 Response failed            : 0
 Response success           : 4
 Retry count                : 0
 Response Ignored           : 0
 Cancelled count            : 0
Response time
 Max Time                   : 100 ms
 Avg Time                   : 10  ms
 Min Time                   : 50  ms
Last transaction
 Transaction Id             : 9
 Transaction type           : Get key
 Transaction status         : Response data received, successfully
 Http code                  : 200 OK (200)
```