MACsec Encryption

- MACsec Encryption, on page 1
- MACsec Key Agreement, on page 2
- MACsec MKA using Certificate-based MACsec, on page 6
- Switch-to-Switch MKA MACsec Must Secure Policy, on page 7
- MKA/MACsec for Port Channel, on page 7
- MACsec Cipher Announcement, on page 7
- How to Configure MACsec Encryption, on page 8
- Additional References for MACsec Encryption, on page 51
- Feature History for MACsec Encryption, on page 52
- Overview of PQC for MACsec with EAP-TLS, on page 52
- Key elements of PQC, on page 53
- PQC implementation in EAP-TLS, on page 54
- Prerequisites, on page 56
- Restrictions, on page 56
- Configuration matrix for MACsec with EAP-TLS, on page 56
- Workflow to set up PQC for MACsec with EAP-TLS, on page 57
- Configure EAP-TLS 1.3, on page 57
- Configure PQC for EAP-TLS sessions, on page 58
- Verify PQC for EAP-TLS sessions, on page 58

MACsec Encryption

MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. Catalyst switches support 802.1AE encryption with MACsec Key Agreement (MKA) on switch-to-host links for encryption between the switch and host device. The switch also supports MACsec encryption for switch-to-switch (inter-network device) security using MKA-based key exchange protocol.

Note When switch-to-switch MACSec is enabled, all traffic is encrypted, except the EAP-over-LAN (EAPOL) packets.

Link layer security can include both packet authentication between switches and MACsec encryption between switches (encryption is optional).

Table 1: MACsec Support on Switch Ports

Connections	MACsec support
Switch-to-host	MACsec MKA encryption
Switch-to-switch	MACsec MKA encryption

Cisco TrustSec is meant only for switch-to-switch links and is not supported on switch ports connected to end hosts, such as PCs or IP phones. MKA is supported on switch-to-host facing links as well as switch-to-switch links. Host-facing links typically use flexible authentication ordering for handling heterogeneous devices with or without IEEE 802.1x, and can optionally use MKA-based MACsec encryption. Network Edge Access Topology (NEAT) is used for compact switches to extend security outside the wiring closet.

MACsec Key Agreement

MACsec, defined in 802.1AE, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. MKA and MACsec are implemented after successful authentication using certificate-based MACsec or Pre Shared Key (PSK) framework.

A switch using MACsec accepts either MACsec or non-MACsec frames, depending on the policy associated with the MKA peer. MACsec frames are encrypted and protected with an integrity check value (ICV). When the switch receives frames from the MKA peer, it decrypts them and calculates the correct ICV by using session keys provided by MKA. The switch compares that ICV to the ICV within the frame. If they are not identical, the frame is dropped. The switch also encrypts and adds an ICV to any frames sent over the secured port (the access point used to provide the secure MAC service to a MKA peer) using the current session key.

The MKA Protocol manages the encryption keys used by the underlying MACsec protocol. The basic requirements of MKA are defined in 802.1x-REV. The MKA Protocol extends 802.1x to allow peer discovery with confirmation of mutual authentication and sharing of MACsec secret keys to protect data exchanged by the peers.

The EAP framework implements MKA as a newly defined EAP-over-LAN (EAPOL) packet. EAP authentication produces a master session key (MSK) shared by both partners in the data exchange. Entering the EAP session ID generates a secure connectivity association key name (CKN). The switch acts as the authenticator for both uplink and downlink; and acts as the key server for downlink. It generates a random secure association key (SAK), which is sent to the client partner. The client is never a key server and can only interact with a single MKA entity, the key server. After key derivation and generation, the switch sends periodic transports to the partner at a default interval of 2 seconds.

The packet body in an EAPOL Protocol Data Unit (PDU) is referred to as a MACsec Key Agreement PDU (MKPDU). MKA sessions and participants are deleted when the MKA lifetime (6 seconds) passes with no MKPDU received from a participant. For example, if a MKA peer disconnects, the participant on the switch continues to operate MKA until 6 seconds have elapsed after the last MKPDU is received from the MKA peer.

Note Integrity check value (ICV) indicator in MKPDU is optional. ICV is not optional when the traffic is encrypted.

EAPoL Announcements indicate the use of the type of keying material. The announcements can be used to announce the capability of the supplicant as well as the authenticator. Based on the capability of each side, the largest common denominator of the keying material could be used.

MKA Policies

To enable MKA on an interface, a defined MKA policy should be applied to the interface. You can configure these options:

- Policy name, not to exceed 16 ASCII characters.
- Confidentiality (encryption) offset of 0, 30, or 50 bytes for each physical interface

Definition of Policy-Map Actions

This section describes the policy-map actions and its definition:

- Activate: Applies a service template to the session.
- Authenticate: Starts authentication of the session.
- Authorize: Explicitly authorizes a session.
- Set-domain: Explicitly sets the domain of a client.
- Terminate: Terminates the method that is running, and deletes all the method details associated with the session.
- Deactivate: Removes the service-template applied to the session. If not applied, no action is taken.
- Set-timer: Starts a timer and gets associated with the session. When the timer expires, any action that needs to be started can be processed.
- Authentication-restart: Restarts authentication.
- Clear-session: Deletes a session.
- Pause: Pauses authentication.

Rest of the actions as self-explanatory and are associated with authentication.

Virtual Ports

Use virtual ports for multiple secured connectivity associations on a single physical port. Each connectivity association (pair) represents a virtual port. In uplink, you can have only one virtual port per physical port. You cannot simultaneously host secured and unsecured sessions in the same VLAN on the same port. Because of this limitation, 802.1x multiple authentication mode is not supported.

The exception to this limitation is in multiple-host mode when the first MACsec supplicant is successfully authenticated and connected to a hub that is connected to the switch. A non-MACsec host connected to the hub can send traffic without authentication because it is in multiple-host mode. We do not recommend using multi-host mode because after the first successful client, authentication is not required for other clients.

Virtual ports represent an arbitrary identifier for a connectivity association and have no meaning outside the MKA Protocol. A virtual port corresponds to a separate logical port ID. Valid port IDs for a virtual port are

0x0002 to 0xFFFF. Each virtual port receives a unique secure channel identifier (SCI) based on the MAC address of the physical interface concatenated with a 16-bit port ID.

MKA Statistics

Some MKA counters are aggregated globally, while others are updated both globally and per session. You can also obtain information about the status of MKA sessions. See Displaying MKA Statistics for further information.

Key Lifetime and Hitless Key Rollover

A MACsec key chain can have multiple pre-shared keys (PSK) each configured with a key id and an optional lifetime. A key lifetime specifies at which time the key expires. In the absence of a lifetime configuration, the default lifetime is unlimited. When a lifetime is configured, MKA rolls over to the next configured pre-shared key in the key chain after the lifetime is expired. Time zone of the key can be local or UTC. Default time zone is UTC.

You can Key rolls over to the next key within the same key chain by configuring a second key in the key chain and configuring a lifetime for the first key. When the lifetime of the first key expires, it automatically rolls over to the next key in the list. If the same key is configured on both sides of the link at the same time, then the key rollover is hitless, that is, key rolls over without traffic interruption.

On all participating devices, the MACsec key chain must be synchronised by using Network Time Protocol (NTP) and the same time zone must be used. If all the participating devices are not synchronized, the connectivity association key (CAK) rekey will not be initiated on all the devices at the same time.

Note The lifetime of the keys need to be overlapped in order to achieve hitless key rollover.

Default timer-based SAK rekey

A Secure Association Key (SAK) is a core security feature in MACsec (Media Access Control Security) that:

- delivers encryption and authentication for Ethernet frames, ensuring secure data transmission, and
- is dynamically created and managed as secure associations are established between network devices.

Cisco Catalyst IE9300 Rugged Series Switches support default timer-based rekey, with rekey frequencies based on interface speed:

- 1 Gbps: approximately 30 minutes
- 10 Gbps: approximately 3 minutes
- 40 Gbps: approximately 45 seconds

Note The Cisco Catalyst IE9300 Rugged Series Switches doesn't support packet numbering (PN) exhaustion rekey.

Replay Protection Window Size

Replay protection is a feature provided by MACsec to counter replay attacks. Each encrypted packet is assigned a unique sequence number and the sequence is verified at the remote end. Frames transmitted through a Metro Ethernet service provider network are highly susceptible to reordering due to prioritization and load balancing mechanisms used within the network.

A replay window is necessary to support the use of MACsec over provider networks that reorder frames. Frames within the window can be received out of order, but are not replay protected. The default window size is 0, which enforces strict reception ordering. The replay window size can be configured in the range of 0 to 2^{32}- 1.

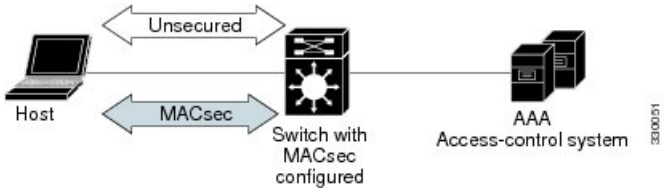
MACsec, MKA and 802.1x Host Modes

You can use MACsec and the MKA Protocol with 802.1x single-host mode, multi-host mode, or Multi Domain Authentication (MDA) mode. Multiple authentication mode is not supported.

Single-Host Mode

The figure shows how a single EAP authenticated session is secured by MACsec by using MKA

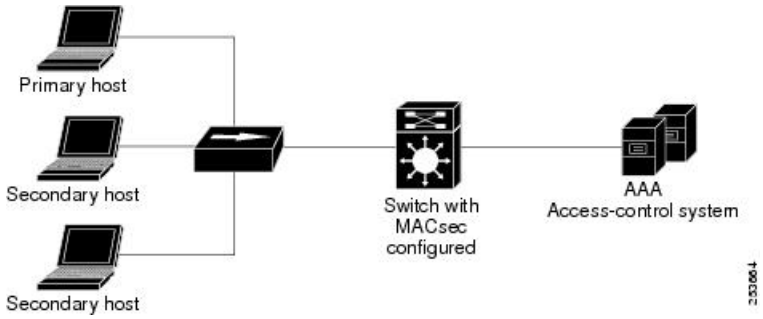
Figure 1: MACsec in Single-Host Mode with a Secured Data Session



Multiple-Host Mode

In standard (not 802.1x REV) 802.1x multiple-host mode, a port is open or closed based on a single authentication. If one user, the primary secured client services client host, is authenticated, the same level of network access is provided to any host connected to the same port. If a secondary host is a MACsec supplicant, it cannot be authenticated and traffic would not flow. A secondary host that is a non-MACsec host can send traffic to the network without authentication because it is in multiple-host mode. The figure shows MACsec in Standard Multiple-Host Unsecure Mode.

Figure 2: MACsec in Multiple-Host Mode - Unsecured



Note Multi-host mode is not recommended because after the first successful client, authentication is not required for other clients, which is not secure.

In standard (not 802.1x REV) 802.1x multiple-domain mode, a port is open or closed based on a single authentication. If the primary user, a PC on data domain, is authenticated, the same level of network access is provided to any domain connected to the same port. If a secondary user is a MACsec supplicant, it cannot be authenticated and traffic would no flow. A secondary user, an IP phone on voice domain, that is a non-MACsec host, can send traffic to the network without authentication because it is in multiple-domain mode.

Multiple-Domain Mode

In standard (not 802.1x REV) 802.1x multiple-domain mode, a port is open or closed based on a single authentication. If the primary user, a PC on data domain, is authenticated, the same level of network access is provided to any domain connected to the same port. If a secondary user is a MACsec supplicant, it cannot be authenticated and traffic would no flow. A secondary user, an IP phone on voice domain, that is a non-MACsec host, can send traffic to the network without authentication because it is in multiple-domain mode.

MACsec MKA using Certificate-based MACsec

MACsec MKA is supported on switch-to-switch links. Using certificate-based MACsec, you can configure MACsec MKA between device uplink ports. Certificate-based MACsec allows mutual authentication and obtains an MSK (master session key) from which the connectivity association key (CAK) is derived for MKA operations. Device certificates are carried, using certificate-based MACsec, for authentication to the AAA server.

Note Certificate-based MACsec is supported for Cisco Catalyst ESS9300 Embedded Series Switch beginning with the Cisco IOS XE 17.13.1 release.

Prerequisites for MACsec MKA Using Certificate-based MACsec

- Ensure that you have a Certificate Authority (CA) server configured for your network.
- Generate a CA certificate.
- Ensure that you have configured Cisco Identity Services Engine (ISE) Release 2.0.
- Ensure that both the participating devices, the CA server, and Cisco Identity Services Engine (ISE) are synchronized using Network Time Protocol (NTP). If time is not synchronized on all your devices, certificates will not be validated.
- Ensure that 802.1x authentication and AAA are configured on your device.

Switch-to-Switch MKA MACsec Must Secure Policy

Must-secure support is enabled on both the ingress and the egress. Must-secure is supported for MKA. With must-secure enabled, only EAPoL traffic will not be encrypted. The rest of the traffic will be encrypted. Unencrypted packets are dropped.

Note Must-secure mode is enabled by default.

MKA/MACsec for Port Channel

MKA/MACsec can be configured on the port members of a port channel. MKA/MACsec is agnostic to the port channel since the MKA session is established between the port members of a port channel.

Note Port channel is supported for PSK-based MACsec but not for certificate-based MACsec.

Note EtherChannel links that are formed as part of the port channel can either be congruent or disparate. That is, the links can either be MACsec-secured or non-MACsec-secured. MKA session between the port members is established even if a port member on one side of the port channel is not configured with MACsec.

We recommend that you enable MKA/MACsec on all the member ports for better security of the port channel.

MACsec Cipher Announcement

Cipher Announcement allows the supplicant and the authenticator to announce their respective MACsec Cipher Suite capabilities to each other. Both the supplicant and the authenticator calculate the largest common supported MACsec Cipher Suite and use the same as the keying material for the MKA session.

Note Only the MACsec Cipher Suite capabilities which are configured in the MKA policy are announced from the authenticator to the supplicant.

There are two types of EAPoL Announcements:

- Unsecured Announcements (EAPoL PDUs) : Unsecured announcements are EAPoL announcements carrying MACsec Cipher Suite capabilities in an unsecured manner. These announcements are used to decide the width of the key used for MKA session prior to authentication.
- Secure Announcements (MKPDUs) : Secure announcements revalidate the MACsec Cipher Suite capabilities which were shared previously through unsecure announcements.

Once the session is authenticated, peer capabilities which were received through EAPoL announcements are revalidated with the secure announcements. If there is a mismatch in the capabilities, the MKA session tears down.

Limitations for MACsec Cipher Announcement

- MACsec Cipher Announcement is supported only on the switch-to-host links.
- The MKA session between the supplicant and the authenticator does not tear down even if the MACsec Cipher Suite capabilities configured on both do not result in a common cipher suite.
- Host to Switch MKA MACsec 256-bit Cipher is not supported on both Network Advantage and Network Essentials.

How to Configure MACsec Encryption

Prerequisites for MACsec Encryption

Prerequisites for MACsec Encryption

- Enable the **ssci-based-on-sci** command while configuring MACsec encryption on the device to allow interoperability with non-Cisco and non-IOS XE devices.
- Ensure that 802.1x authentication and AAA are configured on your device.

Prerequisites for Certificate-Based MACsec

- Ensure that you have a Certificate Authority (CA) server configured for your network.
- Generate a CA certificate.
- Ensure that you have configured Cisco Identity Services Engine (ISE) Release 2.0.
- Ensure that both the participating devices, the CA server, and Cisco Identity Services Engine (ISE) are synchronized using Network Time Protocol (NTP). If time is not synchronized on all your devices, certificates will not be validated.

Restrictions for MACsec Encryption

- MACsec Key Agreement (MKA) is not supported with both stateful and stateless high availability.
- MACsec with MKA is supported only on point-to-point links.
- MACsec configuration is not supported on EtherChannel ports. Instead, MACsec configuration can be applied on the individual member ports of an EtherChannel. To remove MACsec configuration, you must first unbundle the member ports from the EtherChannel, and then remove it from the individual member ports.

- Cisco Catalyst IE9300 Rugged Series Switches support 128-bit MACsec encryption with a Network Essentials license and 256-bit MACsec encryption with a Network Advantage license.
- Certificate-based MACsec is supported only if the access-session is configured as closed or in multiple-host mode. None of the other configuration modes are supported.
- Packet number exhaustion rekey is not supported.
- If the **dot1q tag vlan native** command is configured globally, the dot1x reauthentication will fail on trunk ports.
- MACsec with Precision Time Protocol (PTP) is not supported.
- The **should-secure** access mode is supported on switch-to-switch ports only using PSK authentication.
- PSK fallback key chain is not supported for point-to-multipoint cases.
- PSK fallback key chain is not supported on a high availability setup.
- PSK fallback key chain supports infinite lifetime with one key only.
- The connectivity association key name (CKN) ID used in the fallback key chain must not match any of the CKN IDs used in the primary key chain.
- The following limitations apply only to certificate-based MACsec.
  - The port should be in access mode or trunk mode.
  - MKA is not supported on port channels.
  - Ports with no switch port are not supported.

Recommendations for MACsec Encryption

This section list the recommendations for configuring MACsec encryption:

- Use the confidentiality (encryption) offset as 0 in switch-to-host connections.
- Execute the **shutdown** command, and then the **no shutdown** command on a port, after changing any MKA policy or MACsec configuration for active sessions, so that the changes are applied to active sessions.
- Set the connectivity association key (CAK) rekey overlap timer to 30 seconds or more.

MKA and MACsec Configuration

MACsec is disabled by default. No MKA policies are configured.

Configure an MKA Policy

Perform these steps to configure MACsec Key Agreement (MKA) policy.

Procedure

Step 1 Accessing and creating the MKA policy

a) Use the **enable** command to enable privileged EXEC mode.

Example:

```
Device> enable
```

Enter your password if prompted.

b) Use the **configure terminal** command to enter global configuration mode.

Example:

```
Device# configure terminal
```

c) Use the **mka policy** *policy-name* command to identifies an MKA policy, and enters into the MKA policy configuration mode.

Example:

```
Device(config)# mka policy mka_policy
```

The maximum policy name length is 16 characters.

Note
The default MACsec cipher suite in the MKA policy will always be "GCM-AES-128". If the device supports both "GCM-AES-128" and "GCM-AES-256" ciphers, it is highly recommended to define and use a user defined MKA policy to include both 128 and 256 bits ciphers or only 256 bits cipher, as may be required.

Step 2 Configuring MKA policy parameters

a) Use the **key-server** command to configure MKA key server options.

Example:

```
Device(config-mka-policy)# key-server priority 200
```

It sets priority between 0-255.

Note
When value of key server priority is set to 255, the peer can not become the key server. The key server priority value is valid only for MKA PSK; and not for MKA EAPTLS.

b) Use **sak-rekey interval** *time-interval* to configure the SAK rekey interval in seconds.

Example:

```
Device(config-mka-policy)# sak-rekey interval 50
```

This configuration is optional, with a range from 30 to 65,535.

By default, the SAK rekey interval is set automatically based on interface speed. Any user-defined value must be less than the default interval.

Use the **no** form of this command to stop the SAK rekey timer.

c) Use the **include-icv-indicator** command to enables the ICV indicator in MKPDU.

Example:

```
Device(config-mka-policy)# include-icv-indicator
```

Use the **no** form of this command to disable the ICV indicator.

d) **macsec-cipher-suite** co mand to configure a cipher suite for deriving SAK with 128-bit or 256-bit encryption.

Example:

```
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
```

The syntax includes: **macsec-cipher-suite** {*gcm-aes-128* | *gcm-aes-256*}

e) Use the **confidentiality-offset** command to set the confidentiality (encryption) offset for each physical interface.

Example:

```
Device(config-mka-policy)# confidentiality-offset 0
```

Note
Offset Value can be 0, 30 or 50. If you are using Anyconnect on the client, it is recommended to use Offset 0.

f) Use the **ssci-based-on-sci** command to computes Short Secure Channel Identifier (SSCI) value.

Example:

```
Device(config-mka-policy)# ssci-based-on-sci
```

(Optional) SSCI computation value is based on Secure Channel Identifier (SCI) value. The higher the SCI value, the lower is the SSCI value.

g) Use the **end** command to exit MKA policy configuration mode and return to privileged EXEC mode.

Example:

```
Device(config-mka-policy)# end
```

h) (Optional) Use the **show mka policy** command to display MKA policy configuration information.

Example:

```
Device# show mka policy
```

Configure Switch-to-host MACsec Encryption

Follow these steps to configure MACsec on an interface with one MACsec session for voice and one for data:

SUMMARY STEPS

1. **enable**
2. **configureterminal**

3. **interface** *type number*
4. **switchport access vlan***vlan-id*
5. **switchport mode access**
6. **macsec**
7. **authentication event linksec fail action authorize vlan** *vlan-id*
8. **authentication host-mode multi-domain**
9. **authentication linksec policy must-secure**
10. **authentication port-control auto**
11. **authentication periodic**
12. **authentication timer reauthenticate**
13. **authentication violation protect**
14. **mka policy** *policy-name*
15. **dot1x pae authenticator**
16. **spanning-tree portfast**
17. **end**
18. **show authentication session interface** *interface-id*
19. **show mka sessions**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. • Enter the password if prompted.
Step 2	**configureterminal** **Example:** `Device>configure terminal`	Enters the global configuration mode.
Step 3	**interface** *type number* **Example:** `Device(config)# interface GigabitEthernet 1/0/1`	Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.
Step 4	**switchport access vlan***vlan-id* **Example:** `Device(config-if)# switchport access vlan 1`	Configures the access VLAN for the port.
Step 5	**switchport mode access** **Example:** `Device(config-if)# switchport mode access`	Configures the interface as an access port.

	Command or Action	Purpose
Step 6	**macsec** **Example:** `Device(config-if)# macsec`	Enables 802.1ae MACsec on the interface. The macsec command enables MKA MACsec on switch-to-host links only.
Step 7	**authentication event linksec fail action authorize vlan** *vlan-id* **Example:** `Device(config-if)# authentication event linksec fail action authorize vlan 1`	(Optional) Specifies that the switch processes authentication link-security failures resulting from unrecognized user credentials by authorizing a restricted VLAN on the port after a failed authentication attempt.
Step 8	**authentication host-mode multi-domain** **Example:** `Device(config-if)# authentication host-mode multi-domain`	Configures authentication manager mode on the port to allow both a host and a voice device to be authenticated on the 802.1x-authorized port. If not configured, the default host mode is single.
Step 9	**authentication linksec policy must-secure** **Example:** `Device(config-if)# authentication linksec policy must-secure`	Sets the LinkSec security policy to secure the session with MACsec if the peer is available. If not set, the default is *should secure*.
Step 10	**authentication port-control auto** **Example:** `Device(config-if)# authentication port-control auto`	Enables 802.1x authentication on the port. The port changes to the authorized or unauthorized state based on the authentication exchange between the switch and the client.
Step 11	**authentication periodic** **Example:** `Device(config-if)# authentication periodic`	(Optional) Enables or disables re-authentication for this port .
Step 12	**authentication timer reauthenticate** **Example:** `Device(config-if)# authentication timer reauthenticate`	(Optional) Enters a value between 1 and 65535 (in seconds). Obtains re-authentication timeout value from the server. Default re-authentication time is 3600 seconds.
Step 13	**authentication violation protect** **Example:** `Device(config-if)# configure terminal`	Configures the port to drop unexpected incoming MAC addresses when a new device connects to a port or when a device connects to a port after the maximum number of devices are connected to that port. If not configured, the default is to shut down the port.
Step 14	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Applies an existing MKA protocol policy to the interface, and enable MKA on the interface. If no MKA policy was configured (by entering the**mka policy** global configuration command).
Step 15	**dot1x pae authenticator** **Example:**	Configures the port as an 802.1x port access entity (PAE) authenticator.

	Command or Action	Purpose
	`Device(config-if)# dot1x pae authenticator`	
Step 16	**spanning-tree portfast** **Example:** `Device(config-if)# spanning-tree portfast`	Enables spanning tree Port Fast on the interface in all its associated VLANs. When the Port Fast feature is enabled, the interface changes directly from a blocking state to a forwarding state without making the intermediate spanning-tree state changes
Step 17	**end** **Example:** `Device(config)# end`	Exits interface configuration mode and returns to privileged EXEC mode.
Step 18	**show authentication session interface** *interface-id* **Example:** `Device# show authentication session interface GigabitEthernet 1/0/1`	Verifies the authorized session security status.
Step 19	**show mka sessions** **Example:** `Device# show mka sessions`	Verifies the established MKA sessions.

Configure MACsec MKA using PSK

Beginning in privileged EXEC mode, follow these steps to configure MACsec MKA policies using a Pre Shared Key (PSK).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **key chain** *key-chain-name* **macsec**
4. **key** *hex-string*
5. **cryptographic-algorithm** *{aes-128-cmac | aes-256-cmac}*
6. **key-string** *{ [0|6|7] pwd-string | pwd-string}*
7. **lifetime local** [*start timestamp {hh::mm::ss | day | month | year}*] [**duration** *seconds* | *end timestamp {hh::mm::ss | day | month | year}*]
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**key chain** *key-chain-name* **macsec** **Example:** `Device(config)# key chain keychain1 macsec`	Configures a key chain and enters the key chain configuration mode.
Step 4	**key** *hex-string* **Example:** `Device(config-key-chain)# key 1000`	Configures a unique identifier for each key in the keychain and enters the keychain's key configuration mode. **Note** For 128-bit encryption, use any value between 1 and 32 hex digit key-string. For 256-bit encryption, use 64 hex digit key-string.
Step 5	**cryptographic-algorithm** *{aes-128-cmac | aes-256-cmac}* **Example:** `Device(config-key-chain)# cryptographic-algorithm aes-128-cmac`	Set cryptographic authentication algorithm with 128-bit or 256-bit encryption.
Step 6	**key-string** *{ [0|6|7] pwd-string | pwd-string}* **Example:** `Device(config-key-chain)# key-string 12345678901234567890123456789012`	Sets the password for a key string. Only hex characters must be entered.
Step 7	**lifetime local** [*start timestamp {hh::mm::ss | day | month | year}*] [**duration** *seconds* | *end timestamp {hh::mm::ss | day | month | year}*] **Example:** `Device(config-key-chain)# lifetime local 12:12:00 July 28 2016 12:19:00 July 28 2016`	Sets the lifetime of the pre shared key.
Step 8	**end** **Example:** `Device(config-key-chain)# end`	Exits key chain configuration mode and returns to privileged EXEC mode.

Configure MACsec MKA on an Interface using PSK

Beginning in privileged EXEC mode, follow these steps to configure MACsec MKA policies on an interface using a Pre Shared Key (PSK).

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface-id*

4. **macsec network-link**
5. **mka policy** *policy-name*
6. **mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*]
7. **macsec replay-protection window-size** *frame number*
8. **end**

DETAILED STEPS

Procedure

	Command or Action	Purpose
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	**interface** *interface-id* **Example:** `Device(config-if)# interface GigabitEthernet 0/0/0`	Enters interface configuration mode.
Step 4	**macsec network-link** **Example:** `Device(config-if)# macsec network-link`	Enables MACsec on the interface.
Step 5	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Configures an MKA policy.
Step 6	**mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*] **Example:** `Device(config-if)# mka pre-shared-key key-chain key-chain-name`	Configures an MKA pre-shared-key key-chain name.
Step 7	**macsec replay-protection window-size** *frame number* **Example:** `Device(config-if)# macsec replay-protection window-size 10`	Sets the MACsec window size for replay protection.
Step 8	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

What to do next

It is not recommended to change the MKA policy on an interface with MKA PSK configured when the session is running. However, if a change is required, you must reconfigure the policy as follows:

1. Disable the existing session by removing macsec network-link configuration on each of the participating node using the **no macsec network-link** command
2. Configure the MKA policy on the interface on each of the participating node using the **mka policy policy-name** command.
3. Enable the new session on each of the participating node by using the **macsec network-link** command.

Configuring MACsec MKA Using Certificate-based MACsec

To configure MACsec with MKA on point-to-point links, perform these tasks:

- Configure Certificate Enrollment
  - Generate Key Pairs
  - Configure SCEP Enrollment
  - Configure Certificates Manually
- Configure an Authentication Policy
- Configure certificate-based MACsec Profiles and IEEE 802.1x Credentials
- Configure MKA MACsec using certificate-based MACsec on Interfaces

Generate Key Pairs

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa label** *label-name* **general-keys modulus** *size*
4. **end**
5. **show authentication session interface** *interface-id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa label** *label-name* **general-keys modulus** *size* **Example:** `Device(config)# crypto key generate rsa label general-keys modulus 2048`	Generates a RSA key pair for signing and encryption. You can also assign a label to each key pair using the label keyword. The label is referenced by the trustpoint that uses the key pair. If you do not assign a label, the key pair is automatically labeled <Default-RSA-Key>. If you do not use additional keywords this command generates one general purpose RSA key pair. If the modulus is not specified, the default key modulus of 1024 is used. You can specify other modulus sizes with the modulus keyword.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 5	**show authentication session interface** *interface-id* **Example:** `Device# show authentication session interface gigabitethernet 0/1/1`	Verifies the authorized session security status.

Configure Enrollment using SCEP

Simple Certificate Enrollment Protocol (SCEP) is a Cisco-developed enrollment protocol that uses HTTP to communicate with the certificate authority (CA) or registration authority (RA). SCEP is the most commonly used method for sending and receiving requests and certificates.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *server name*
4. **enrollment url** *url name pem*
5. **rsakeypair** *label*
6. **serial-number none**
7. **ip-address none**
8. **revocation-check crl**
9. **auto-enroll** *percent* **regenerate**
10. **exit**
11. **crypto pki authenticate** *name*
12. **end**
13. **show crypto pki certificate** *trustpoint name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
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	**crypto pki trustpoint** *server name* **Example:** `Device(config)# crypto pki trustpoint ka`	Declares the trustpoint and a given name and enters ca-trustpoint configuration mode.
Step 4	**enrollment url** *url name pem* **Example:** `Device(ca-trustpoint)# enrollment url http://url:80`	Specifies the URL of the CA on which your device should send certificate requests. An IPv6 address can be added in the URL enclosed in brackets. For example: http:// [2001:DB8:1:1::1]:80. The pem keyword adds privacy-enhanced mail (PEM) boundaries to the certificate request.
Step 5	**rsakeypair** *label* **Example:** `Device(ca-trustpoint)# rsakeypair exampleCAkeys`	Specifies which key pair to associate with the certificate. **Note** The **rsakeypair** name must match the trust-point name.
Step 6	**serial-number none** **Example:** `Device(ca-trustpoint)# serial-number none`	The **none** keyword specifies that a serial number will not be included in the certificate request.
Step 7	**ip-address none** **Example:** `Device(ca-trustpoint)# ip-address none`	The **none** keyword specifies that no IP address should be included in the certificate request.
Step 8	**revocation-check crl** **Example:** `Device(ca-trustpoint)# revocation-check crl`	Specifies CRL as the method to ensure that the certificate of a peer has not been revoked.
Step 9	**auto-enroll** *percent* **regenerate** **Example:** `Device(ca-trustpoint)# auto-enroll 90 regenerate`	Enables auto-enrollment, allowing the client to automatically request a rollover certificate from the CA. If auto-enrollment is not enabled, the client must be manually re-enrolled in your PKI upon certificate expiration.

	Command or Action	Purpose
		By default, only the Domain Name System (DNS) name of the device is included in the certificate. Use the percent argument to specify that a new certificate will be requested after the percentage of the lifetime of the current certificate is reached. Use the regenerate keyword to generate a new key for the certificate even if a named key already exists. If the key pair being rolled over is exportable, the new key pair will also be exportable. The following comment will appear in the trustpoint configuration to indicate whether the key pair is exportable: "! RSA key pair associated with trustpoint is exportable." It is recommended that a new key pair be generated for security reasons.
Step 10	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 11	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate myca`	Retrieves the CA certificate and authenticates it.
Step 12	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.
Step 13	**show crypto pki certificate** *trustpoint name* **Example:** `Device# show crypto pki certificate ka`	Displays information about the certificate for the trust point.

Configure Enrollment Manually

If your CA does not support SCEP or if a network connection between the router and CA is not possible. Perform the following task to set up manual certificate enrollment:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *server name*
4. **enrollment url** *url name pem*
5. **rsakeypair** *label*
6. **serial-number none**
7. **ip-address none**

8. **revocation-check crl**
9. **exit**
10. **crypto pki authenticate** *name*
11. **crypto pki enroll** *name*
12. **crypto pki import** *name* **certificate**
13. **end**
14. **show crypto pki certificate** *trustpoint name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
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	**crypto pki trustpoint** *server name* **Example:** `Device# crypto pki trustpoint ka`	Declares the trustpoint and a given name and enters ca-trustpoint configuration mode.
Step 4	**enrollment url** *url name pem* **Example:** `Device(ca-trustpoint)# enrollment url http://url:80`	Specifies the URL of the CA on which your device should send certificate requests. An IPv6 address can be added in the URL enclosed in brackets. For example: http:// [2001:DB8:1:1::1]:80. The pem keyword adds privacy-enhanced mail (PEM) boundaries to the certificate request.
Step 5	**rsakeypair** *label* **Example:** `Device(ca-trustpoint)#  rsakeypair exampleCAkeys`	Specifies which key pair to associate with the certificate.
Step 6	**serial-number none** **Example:** `Device(ca-trustpoint)# serial-number none`	The **none** keyword specifies that a serial number will not be included in the certificate request.
Step 7	**ip-address none** **Example:** `Device(ca-trustpoint)# ip-address none`	The **none** keyword specifies that no IP address should be included in the certificate request.

	Command or Action	Purpose
Step 8	**revocation-check crl** **Example:** `Device(ca-trustpoint)# revocation-check crl`	Specifies CRL as the method to ensure that the certificate of a peer has not been revoked.
Step 9	**exit** **Example:** `Device(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode and returns to global configuration mode.
Step 10	**crypto pki authenticate** *name* **Example:** `Device(config)# crypto pki authenticate myca`	Retrieves the CA certificate and authenticates it.
Step 11	**crypto pki enroll** *name* **Example:** `Device(config)# crypto pki enroll myca`	Generates certificate request and displays the request for copying and pasting into the certificate server. Enter enrollment information when you are prompted. For example, specify whether to include the device FQDN and IP address in the certificate request. You are also given the choice about displaying the certificate request to the console terminal. The base-64 encoded certificate with or without PEM headers as requested is displayed.
Step 12	**crypto pki import** *name* **certificate** **Example:** `Device(config)# crypto pki import myca certificate`	Imports a certificate via TFTP at the console terminal, which retrieves the granted certificate. The device attempts to retrieve the granted certificate via TFTP using the same filename used to send the request, except the extension is changed from ".req" to ".crt". For usage key certificates, the extensions "-sign.crt" and "-encr.crt" are used. The device parses the received files, verifies the certificates, and inserts the certificates into the internal certificate database on the switch. **Note** Some CAs ignore the usage key information in the certificate request and issue general purpose usage certificates. If your CA ignores the usage key information in the certificate request, only import the general purpose certificate. The router will not use one of the two key pairs generated.
Step 13	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

	Command or Action	Purpose
Step 14	**show crypto pki certificate** *trustpoint name* **Example:** `Device# show crypto pki certificate  ka`	Displays information about the certificate for the trust point.

Enable 802.1x Authentication and Configure AAA

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **aaa new-model**
4. **dot1x system-auth-control**
5. **radius server** *name*
6. **address** *ip_address* **auth-port** *port_number* **acct-port** *port_number*
7. **automate-tester username** *username*
8. **key** *string*
9. **radius-server deadtime** *minutes*
10. **exit**
11. **aaa group server radius** *group_name*
12. **server** *name*
13. **exit**
14. **aaa authentication dot1x default group** *group_name*
15. **aaa authorization network default group** *group_name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
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	**aaa new-model** **Example:** `device(config)# aaa new-model`	Enables AAA.
Step 4	**dot1x system-auth-control** **Example:**	Enables 802.1X on your device.

	Command or Action	Purpose
	`device(config)# dot1x system-auth-control`	
Step 5	**radius server** *name* **Example:** `device(config)# radius server ISE`	Specifies the name of the RADIUS server configuration for Protected Access Credential (PAC) provisioning and enters RADIUS server configuration mode.
Step 6	**address** *ip_address* **auth-port** *port_number* **acct-port** *port_number* **Example:** `device(config-radius-server)# address ipv4 10.64.72.90 auth-port 1645 acct-port 1646`	Configures the IPv4 address for the RADIUS server accounting and authentication parameters.
Step 7	**automate-tester username** *username* **Example:** `device(config-radius-server)# automate-tester username dummy`	Enables the automated testing feature for the RADIUS server. With this practice, the device sends periodic test authentication messages to the RADIUS server. It looks for a Radius response from the server. A success message is not necessary; a failed authentication suffices, because it shows that the server is alive.
Step 8	**key** *string* **Example:** `device(config-radius-server)# key dummy123`	Configures the authentication and encryption key for all RADIUS communications between the device and the RADIUS server.
Step 9	**radius-server deadtime** *minutes* **Example:** `device(config-radius-server)#radius-server deadtime 2`	Improves RADIUS response time when some servers might be unavailable and skips unavailable servers immediately.
Step 10	**exit**	Returns to global configuration mode.
Step 11	**aaa group server radius** *group_name* **Example:** `device(config)# aaa group server radius ISEGRP`	Groups different RADIUS server hosts into distinct lists and distinct methods, and enters server group configuration mode.
Step 12	**server** *name* **Example:** `device(config)#server ise`	
Step 13	**exit** **Example:** `device(config-radius-server)# exit`	Returns to global configuration mode.
Step 14	**aaa authentication dot1x default group** *group_name* **Example:** `device(config)# aaa authentication dot1x default group ISEGRP`	Sets the default authentication server group for IEEE 802.1x.

	Command or Action	Purpose
Step 15	**aaa authorization network default group** *group_name* **Example:** `device(config)# aaa authorization network default group ISEGRP`	Sets the network authorization default group.

Apply the 802.1x MKA MACsec Configuration on the Interfaces

To apply MKA MACsec using EAP-TLS to interfaces, complete the following steps:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *interface_id*
4. **macsec network-link**
5. **authentication periodic**
6. **authentication timer reauthenticate interval**
7. **access-session host-mode multi-domain**
8. **access-session closed**
9. **access-session port-control auto**
10. **dot1x pae both**
11. **dot1x credentials profile**
12. **dot1x supplicant eap profile** *profile_name*
13. **dot1x authenticator eap profile** *profile_name*
14. **service-policy type control subscriber** *control_policy_name*
15. **exit**
16. **show macsec interface**
17. **copy running-config startup-config**

DETAILED STEPS

Procedure

	Command or Action	Purpose
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.

	Command or Action	Purpose
Step 3	**interface** *interface_id* **Example:** `device(config)# interface te0/1/2`	Identifies the MACsec interface, and enter interface configuration mode. The interface must be a physical interface.
Step 4	**macsec network-link** **Example:** `device(config)# macsec network-link`	Enables MACsec on the interface.
Step 5	**authentication periodic** **Example:** `device(config)# authentication periodic`	Enables reauthentication for this port.
Step 6	**authentication timer reauthenticate interval** **Example:** `device(config)# authentication timer reauthenticate interval`	Sets the reauthentication interval.
Step 7	**access-session host-mode multi-domain** **Example:** `device(config)# access-session host-mode multi-domain`	Allows hosts to gain access to the interface.
Step 8	**access-session closed** **Example:** `device(config)# access-session closed`	Prevents preauthentication access on the interface.
Step 9	**access-session port-control auto** **Example:** `device(config)# access-session port-control auto`	Sets the authorization state of a port.
Step 10	**dot1x pae both** **Example:** `device(config)# dot1x pae both`	Configures the port as an 802.1X port access entity (PAE) supplicant and authenticator.
Step 11	**dot1x credentials profile** **Example:** `device(config)# dot1x credentials profile`	Assigns a 802.1x credentials profile to the interface.
Step 12	**dot1x supplicant eap profile** *profile_name* **Example:** `device(config)# dot1x supplicant eap profile eap1`	Assigns the EAP-TLS profile to the interface.
Step 13	**dot1x authenticator eap profile** *profile_name* **Example:**	Assigns the EAP-TLS profile to use during 802.1x authentication.

	Command or Action	Purpose
	`device(config)# dot1x authenticator eap profile eap1`	
Step 14	**service-policy type control subscriber** *control_policy_name* **Example:** `device(config)# service-policy type control subscriber controlPolicy2`	Applies a subscriber control policy to the interface.
Step 15	**exit** **Example:** `device(config)# exit`	Returns to privileged EXEC mode.
Step 16	**show macsec interface** **Example:** `device# show macsec interface`	Displays MACsec details for the interface.
Step 17	**copy running-config startup-config** **Example:** `device# copy running-config startup-config`	(Optional) Saves your entries in the configuration file.

Configure MKA/MACsec for Port Channel using PSK

Beginning in privileged EXEC mode, complete the following steps to configure MKA policies on an interface using a pre-shared key (PSK):

Procedure

	Command or Action	Purpose
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	**interface** *interface-id* **Example:** `Device(config-if)# interface gigabitethernet 1/0/3`	Enters interface configuration mode.
Step 4	**macsec network-link** **Example:** `Device(config-if)# macsec network-link`	Enables MACsec on the interface. Supports layer 2 and layer 3 port channels.

	Command or Action	Purpose
Step 5	**mka policy** *policy-name* **Example:** `Device(config-if)# mka policy mka_policy`	Configures an MKA policy.
Step 6	**mka pre-shared-key key-chain** *key-chain name* [**fallback key-chain** *key-chain name*] **Example:** `Device(config-if)# mka pre-shared-key key-chain key-chain-name`	Configures an MKA pre-shared-key key-chain name. **Note** The MKA pre-shared key can be configured on either physical interface or subinterfaces and not on both.
Step 7	**macsec replay-protection window-size** *frame number* **Example:** `Device(config-if)# macsec replay-protection window-size 0`	Sets the MACsec window size for replay protection.
Step 8	**channel-group** *channel-group-number* **mode** {**auto** | **desirable**} | {**active** | **passive**} | {**on**} **Example:** `Device(config-if)# channel-group 3 mode auto active on`	Configures the port in a channel group and sets the mode. **Note** You cannot configure ports in a channel group without configuring MACsec on the interface. You must configure the commands in Step 3, 4, 5 and 6 before this step. The channel-number range is from 1 to 4096. The port channel that is associated with this channel group is automatically created if the port channel does not already exist. For mode, select one of the following keywords: • **auto**: Enables PAgP only if a PAgP device is detected. This places the port into a passive negotiating state, in which the port responds to PAgP packets it receives but does not start PAgP packet negotiation. **Note** The **auto** keyword is not supported when EtherChannel members are from different switches in the switch stack. • **desirable**: Unconditionally enables PAgP. This places the port into an active negotiating state, in which the port starts negotiations with other ports by sending PAgP packets. **Note** The **desirable** keyword is not supported when EtherChannel members are from different switches in the switch stack. • **on**: Forces the port to channel without PAgP or LACP. In the on mode, an EtherChannel exists only when a

	Command or Action	Purpose
		port group in the **on** mode is connected to another port group in the **on** mode. • **active**: Enables LACP only if a LACP device is detected. It places the port into an active negotiating state in which the port starts negotiations with other ports by sending LACP packets. • **passive**: Enables LACP on the port and places it into a passive negotiating state in which the port responds to LACP packets that it receives, but does not start LACP packet negotiation.
Step 9	**end** **Example:** `Device(config-if)# cend`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure Port Channel Logical Interfaces for Layer 2 EtherChannels

To create a port channel interface for a Layer 2 EtherChannel, perform this task:

Procedure

	Command or Action	Purpose
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	**interface port-channel** *channel-group-number* **Example:** `Device(config)# interface port-channel 1`	Creates the port channel interface. **Note** Use the **no** form of this command to delete the port channel interface.
Step 4	**switchport** **Example:** `Device(config-if)# switchport`	Switches an interface that is in Layer 3 mode into Layer 2 mode for Layer 2 configuration.
Step 5	**switchport mode** {**access** | **trunk**} **Example:** `Device(config-if)# switchport mode access`	Assigns all ports as static-access ports in the same VLAN, or configure them as trunks.

	Command or Action	Purpose
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configure Port Channel Logical Interfaces for Layer 3 EtherChannels

To create a port channel interface for a Layer 3 EtherChannel, perform this task:

Procedure

	Command or Action	Purpose
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	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/2`	Enters interface configuration mode.
Step 4	**no switchport** **Example:** `Device(config-if)# no switchport`	Switches an interface that is in Layer 2 mode into Layer 3 mode for Layer 3 configuration.
Step 5	**ip address** *ip-address subnet_mask* **Example:** `Device(config-if)# ip address 10.2.2.3 255.255.255.254`	Assigns an IP address and subnet mask to the EtherChannel.
Step 6	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring MACsec Cipher Announcement

Configure an MKA Policy for Secure Announcement

Beginning in privileged EXEC mode, follow these steps to create an MKA Protocol policy to enable secure announcement in MKPDUs. By default, secure announcements are disabled.

Procedure

	Command or Action	Purpose
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	**mka policy** *policy-name* **Example:** `Device(config)# mka policy mka_policy`	Identifies an MKA policy and enters MKA policy configuration mode. The maximum policy name length is 16 characters. **Note** The default MACsec cipher suite in the MKA policy is GCM-AES-128. If the device supports both "GCM-AES-128" and "GCM-AES-256" ciphers, it is highly recommended to define and use a user defined MKA policy to include both 128 and 256 bits ciphers or only 256 bits cipher, as may be required.
Step 4	**key-server** *priority* **Example:** `Device(config-mka-policy)# key-server priority 200`	Configures MKA key server options and sets priority between 0-255. **Note** When value of key server priority is set to 255, the peer cannot become the key server. The key server priority value is valid only for MKA PSK. This does not apply to MKA EAP-TLS.
Step 5	**send-secure-announcements** **Example:** `Device(config-mka-policy)# send-secure-announcements`	Enables sending of secure announcements. Use the **no** form of the command to disable sending of secure announcements. By default, secure announcements are disabled.
Step 6	**macsec-cipher-suite** {*gcm-aes-128* | *gcm-aes-256*} **Example:** `Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128`	Configures cipher suite for deriving SAK with 128-bit or 256-bit encryption.
Step 7	**end** **Example:** `Device(config-mka-policy)# end`	Exits MKA policy configuration mode and returns to privileged EXEC mode.
Step 8	**show mka policy** **Example:**	Displays MKA policies.

	Command or Action	Purpose
	`Device# show mka policy`	

Configure Secure Announcement Globally

Beginning in privileged EXEC mode, follow these steps to enable secure announcement globally across all the MKA Policies.

Procedure

	Command or Action	Purpose
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	**mka defaults policy send-secure-announcements** **Example:** `Device(config)# mka defaults policy send-secure-announcements`	Enables sending of secure announcements in MKPDUs across MKA policies. By default, secure announcements are disabled.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Configure EAPoL Announcements on an Interface

Beginning in privileged EXEC mode, follow these steps to configure EAPoL Announcement on an interface.

Procedure

	Command or Action	Purpose
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.

	Command or Action	Purpose
Step 3	**interface** *interface-id* **Example:** `Device(config)# interface gigabitethernet 1/0/1`	Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.
Step 4	**eapol annoucement** **Example:** `Device(config-if)# eapol announcement`	Enables EAPoL announcements. Use the **no** form of the command to disable EAPoL announcements. By default, EAPoL announcements are disabled.
Step 5	**end** **Example:** `Device(config-if)# configure terminal`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for MACsec Encryption

Example: Configuring MKA and MACsec

This example shows how to create an MKA policy:

```
Device> enable
Device# configure terminal
Device(config)# mka policy mka_policy
Device(config-mka-policy)# key-server priority 200
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 30
Device(config-mka-policy)# ssci-based-on-sci
Device(config-mka-policy)#end
```

This example shows how to configure MACsec on an interface:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# switchport access vlan 1
Device(config-if)# switchport mode access
Device(config-if)# macsec
Device(config-if)#access-session event linksec fail action authorize vlan 1
Device(config-if)# access-session host-mode multi-domain
Device(config-if)# access-session linksec policy must-secure
Device(config-if)# access-session port-control auto
Device(config-if)#authentication periodic
Device(config-if)# authentication timer reauthenticate
Device(config-if)# authentication violation protect
Device(config-if)#mka policy mka_policy
Device(config-if)# dot1x pae authenticator
Device(config-if)# spanning-tree portfast
Device(config-if)#end
```

Examples: Configuring MACsec MKA Using PSK

This example shows how to configure MACsec MKA using PSK.

```
Device> enable
Device# configure terminal
Device(config)# Key chain keychain1 macsec
Device(config-keychain)# key 1000
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789012
Device(config-keychain-key)# lifetime local 12:12:00 July 28 2016 12:19:00 July 28 2016
Device(config-keychain-key)# end
```

This example shows how to configure MACsec MKA on an interface using PSK.

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 0/0/0
Device(config-if)# mka policy mka_policy
Device(config-if)# mka pre-shared-key key-chain key-chain-name
Device(config-if)# macsec replay-protection window-size 10
Device(config-if)# end
```

MKA-PSK: CKN Behavior Change

Starting Cisco IOS XE Fuji 16.8.1 release, for MKA PSK sessions, the CKN uses exactly the same string as the CKN which is configured as the hex-string for the key, instead of the fixed 32 bytes.

```
Device> enable
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 11
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

The following is sample output of the **show mka session** command for the above configuration:

```
Device# show mka session

Total MKA Sessions....... 1
Secured Sessions... 1
Pending Sessions... 0
=============================================================================================
Interface      Local-TxSCI         Policy-Name      Inherited         Key-Server

Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN

=============================================================================================
Et0/0          aabb.cc00.6600/0002     icv            NO                NO

2              aabb.cc00.6500/0002 1               Secured           11   *Note that the
 CKN key-string is exactly the same that has been configured for the key as hex-string.*
```

In case of interoperability between two images, where one having the CKN behavior change, and one without the CKN behavior change, the hex-string for the key must be a 64-character hex-string with zero padded for it to work on a device that has an image with the CKN behavior change. See the examples below:

Configuration without CKN key-string behavior change:

```
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 11
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

Configuration with CKN key-string behavior change:

```
Device# configure terminal
Device(config)# key chain abc macsec
Device(config-keychain)# key 11000000000000000000000000000000000000000000000000000000000000000000
Device(config-keychain-key)# cryptographic-algorithm aes-128-cmac
Device(config-keychain-key)# key-string 12345678901234567890123456789013
Device(config-keychain-key)# lifetime local 12:21:00 Sep 9 2015 infinite
Device(config-keychain-key)# end
```

Examples: Configuring MACsec MKA using Certificate-based MACsec

This example shows how to configure MACsec MKA using certificate-based MACsec:

```
Device> enable
Device# configure terminal
Device(config)# interface Gigabitethernet 1/0/1
Device(config-if)# macsec network-link
Device(config-if)# authentication periodic
Device(config-if)# authentication timer reauthenticate interval
Device(config-if)#access-session host-mode multi-domain
Device(config-if)# access-session closed
Device(config-if)# access-session port-control auto
Device(config-if)# dot1x pae both
Device(config-if)#dot1x credentials profile
Device(config-if)# dot1x supplicant eap profile profile_eap_tls
Device(config-if)#service-policy type control subscriber sub1
Device(config-if)# end
```

Examples: Configuring MACsec MKA for Port Channel using PSK

Etherchannel Mode — Static/On

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode on:

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode on
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode on
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the**show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)           -        Te1/0/1(P)  Te1/0/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
```

```
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
 Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


 Number of channel-groups in use: 1
 Number of aggregators:           1

 Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

 2      Po2(RU)           -        Te1/0/1(P)   Te1/0/2(P)
```

Etherchannel Mode — LACP

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode as LACP.

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode active
```

```
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode active
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1


------+-------------+-----------+-----------------------------------------------

2      Po2(SU)         LACP      Te1/1/1(P)  Te1/1/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)          LACP      Te1/1/1(P)   Te1/1/2(P)
```

Etherchannel Mode — PAgP

The following is sample configuration on Device 1 and Device 2 with EtherChannel Mode as PAgP:

```
Device> enable
Device# configure terminal
Device(config)# key chain KC macsec
Device(config-key-chain)# key 1000
Device(config-key-chain)# cryptographic-algorithm aes-128-cmac
Device(config-key-chain)# key-string FC8F5B10557C192F03F60198413D7D45
Device(config-key-chain)# exit
Device(config)# mka policy POLICY
Device(config-mka-policy)# key-server priority 0
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-128
Device(config-mka-policy)# confidentiality-offset 0
```

```
Device(config-mka-policy)# exit
Device(config)# interface gigabitethernet 1/0/1
Device(config-if)# channel-group 2 mode desirable
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# exit
Device(config)# interface gigabitethernet 1/0/2
Device(config-if)# channel-group 2 mode desirable
Device(config-if)# macsec network-link
Device(config-if)# mka policy POLICY
Device(config-if)# mka pre-shared-key key-chain KC
Device(config-if)# end
```

Layer 2 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# switchport
Device(config-if)# switchport mode trunk
Device(config-if)# no shutdown
Device(config-if)# end
```

The following shows a sample output from the **show etherchannel summary** command.

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1


------+-------------+-----------+-----------------------------------------------

2      Po2(SU)          PAgP      Te1/1/1(P)  Te1/1/2(P)
```

Layer 3 EtherChannel Configuration

Device 1

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.3 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

Device 2

```
Device> enable
Device# configure terminal
Device(config)# interface port-channel 2
Device(config-if)# no switchport
Device(config-if)# ip address 10.25.25.4 255.255.255.0
Device(config-if)# no shutdown
Device(config-if)# end
```

The following is sample output from the **show etherchannel summary** command:

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports

------+-------------+-----------+-----------------------------------------------

2      Po2(RU)         PAgP      Te1/1/1(P)  Te1/1/2(P)
```

Displaying Active MKA Sessions

The following shows all the active MKA sessions.

```
Device# show mka sessions interface Te1/0/1

===============================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN
```

```
Te1/0/1         00a3.d144.3364/0025 POLICY           NO                  NO

37              701f.539b.b0c6/0032 1                Secured
1000
```

Examples: Configuring MACsec Cipher Announcement

This example shows how to configure MKA policy for Secure Announcement:

```
Device> enable
Device# configure terminal
Device(config)# mka policy mka_policy
Device(config-mka-policy)# key-server 2
Device(config-mka-policy)# send-secure-announcements
Device(config-mka-policy)#macsec-cipher-suite gcm-aes-128confidentiality-offset 0
Device(config-mka-policy)# end
```

This example shows how to configure Secure Announcement globally:

```
Device> enable
Device# configure terminal
Device(config)# mka defaults policy send-secure-announcements
Device(config)# end
```

This example shows how to configure EAPoL Announcements on an interface:

```
Device> enable
Device# configure terminal
Device(config)# interface GigabitEthernet 1/0/1
Device(config-if)# eapol announcement
Device(config-if)# end
```

The following is sample output from the **show running-config interface** *interface-name* command with EAPoL announcement enabled.

```
Device# show running-config interface GigabitEthernet 1/0/1

switchport mode access
 macsec
 access-session host-mode multi-host
 access-session closed
 access-session port-control auto
 dot1x pae authenticator
 dot1x timeout quiet-period 10
 dot1x timeout tx-period 5
 dot1x timeout supp-timeout 10
 dot1x supplicant eap profile peap
 eapol announcement
 spanning-tree portfast
 service-policy type control subscriber Dot1X
```

The following is sample output from the **show mka sessions interface** *interface-name* **detail** command with secure announcement disabled.

```
Device# show mka sessions interface GigabitEthernet 1/0/1 detail
```

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

Local Tx-SCI..............  204c.9e85.ede4/002b
Interface MAC Address....  204c.9e85.ede4
MKA Port Identifier......  43
Interface Name...........  GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)...  D46CBEC05D5D67594543CEAE
Message Number (MN)......  89567
EAP Role.................  NA
Key Server...............  YES
MKA Cipher Suite.........  AES-128-CMAC

Latest SAK Status........  Rx & Tx
Latest SAK AN............  0
Latest SAK KI (KN).......  D46CBEC05D5D67594543CEAE00000001 (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)

MKA Policy Name..........  p2
Key Server Priority......  2
Delay Protection.........  NO
Replay Protection........  YES
Replay Window Size.......  0
Confidentiality Offset...  0
Algorithm Agility........  80C201
Send Secure Announcement..  DISABLED
SAK Cipher Suite.........  0080C20001000001 (GCM-AES-128)
MACsec Capability........  3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired...........  YES

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

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89555       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

```
Dormant Peers List:
  MI                          MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

The following is sample output from the **show mka sessions details** command with secure announcement disabled.

```
Device# show mka sessions details

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

Local Tx-SCI.............. 204c.9e85.ede4/002b
Interface MAC Address.... 204c.9e85.ede4
MKA Port Identifier...... 43
Interface Name........... GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)... D46CBEC05D5D67594543CEAE
Message Number (MN)...... 89572
EAP Role................. NA
Key Server............... YES
MKA Cipher Suite......... AES-128-CMAC

Latest SAK Status........ Rx & Tx
Latest SAK AN............ 0
Latest SAK KI (KN)....... D46CBEC05D5D67594543CEAE00000001 (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)

MKA Policy Name.......... p2
Key Server Priority...... 2
Delay Protection......... NO
Replay Protection........ YES
Replay Window Size....... 0
Confidentiality Offset... 0
Algorithm Agility........ 80C201
Send Secure Announcement.. DISABLED
SAK Cipher Suite......... 0080C20001000001 (GCM-AES-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

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

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

```
  ----------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89560       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ----------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ----------------------------------------------------------------------
```

The following is sample output from the **show mka policy** *policy-name* **detail** command with secure announcement disabled.

```
Device# show mka policy p2 detail

MKA Policy Configuration ("p2")
=======================
MKA Policy Name........ p2
Key Server Priority.... 2
Confidentiality Offset. 0
Send Secure Announcement..DISABLED
Cipher Suite(s)........ GCM-AES-128

Applied Interfaces...
  GigabitEthernet1/0/1
```

Examples: Displaying MKA Information

The following is sample output from the **show mka sessions** command.

```
Device# show mka sessions


Total MKA Sessions....... 1
      Secured Sessions... 1
      Pending Sessions... 0

=============================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status            CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                YES
43             c800.8459.e764/002a 1                Secured
0100000000000000000000000000000000000000000000000000000000000000
```

The following is sample output from the **show mka sessions interface** *interface-name* command.

```
Device# show mka sessions interface GigabitEthernet 1/0/1

Summary of All Currently Active MKA Sessions on Interface
GigabitEthernet1/0/1...

=============================================================================
```

```
Interface      Local-TxSCI         Policy-Name      Inherited
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status             CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                 YES
43             c800.8459.e764/002a 1                 Secured
01000000000000000000000000000000000000000000000000000000000000000000
```

The following is sample output from the **show mka sessions interface** *interface-name* **detail**command.

```
Device# show mka sessions interface GigabitEthernet 1/0/1 detail


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

Local Tx-SCI.............. 204c.9e85.ede4/002b
Interface MAC Address.... 204c.9e85.ede4
MKA Port Identifier...... 43
Interface Name........... GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
01000000000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)... D46CBEC05D5D67594543CEAE
Message Number (MN)...... 89567
EAP Role................. NA
Key Server............... YES
MKA Cipher Suite......... AES-128-CMAC

Latest SAK Status........ Rx & Tx
Latest SAK AN............ 0
Latest SAK KI (KN)....... D46CBEC05D5D67594543CEAE00000001 (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)

MKA Policy Name.......... p2
Key Server Priority...... 2
Delay Protection......... NO
Replay Protection........ YES
Replay Window Size....... 0
Confidentiality Offset... 0
Algorithm Agility........ 80C201
Send Secure Announcement.. DISABLED
SAK Cipher Suite......... 0080C20001000001 (GCM-AES-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

# of MACsec Capable Live Peers............ 1
```

```
# of MACsec Capable Live Peers Responded.. 1

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89555       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

The following is sample output from the **show mka sessions details** command:

```
Device# show mka sessions details

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

Local Tx-SCI..............  204c.9e85.ede4/002b
Interface MAC Address....  204c.9e85.ede4
MKA Port Identifier......  43
Interface Name...........  GigabitEthernet1/0/1
Audit Session ID.........
CAK Name (CKN)...........
0100000000000000000000000000000000000000000000000000000000000000
Member Identifier (MI)...  D46CBEC05D5D67594543CEAE
Message Number (MN)......  89572
EAP Role.................  NA
Key Server...............  YES
MKA Cipher Suite.........  AES-128-CMAC

Latest SAK Status........  Rx & Tx
Latest SAK AN............  0
Latest SAK KI (KN).......  D46CBEC05D5D67594543CEAE00000001 (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)

MKA Policy Name..........  p2
Key Server Priority......  2
Delay Protection.........  NO
Replay Protection........  YES
Replay Window Size.......  0
Confidentiality Offset...  0
Algorithm Agility........  80C201
```

```
Send Secure Announcement.. DISABLED
SAK Cipher Suite......... 0080C20001000001 (GCM-AES-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

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

Live Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
  38046BA37D7DA77E06D006A9  89560       c800.8459.e764/002a   10

Potential Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------

Dormant Peers List:
  MI                        MN          Rx-SCI (Peer)        KS Priority
  ------------------------------------------------------------------------
```

The following is sample output from the **show mka policy** command:

```
Device# show mka policy


MKA Policy Summary...

Policy            KS       Delay   Replay  Window Conf   Cipher
 Interfaces
Name              Priority Protect Protect Size   Offset Suite(s)
 Applied
================================================================================
*DEFAULT POLICY*  0        FALSE   TRUE    0      0      GCM-AES-128

p1                1        FALSE   TRUE    0      0      GCM-AES-128

p2                2        FALSE   TRUE    0      0      GCM-AES-128
 Gi1/0/1
```

The following is sample output from the **show mka policy** *policy-name* command:

```
Device# show mka policy p2


MKA Policy Summary...

Policy            KS       Delay   Replay  Window Conf   Cipher
 Interfaces
Name              Priority Protect Protect Size   Offset Suite(s)
 Applied
================================================================================
p2                2        FALSE   TRUE    0      0      GCM-AES-128
 Gi1/0/1
```

The following is sample output from the **show mka policy** *policy-name* **detail** command:

```
Device# show mka policy p2 detail

MKA Policy Configuration ("p2")
==========================
MKA Policy Name........ p2
Key Server Priority.... 2
Confidentiality Offset. 0
Send Secure Announcement..DISABLED
Cipher Suite(s)........ GCM-AES-128

Applied Interfaces...
  GigabitEthernet1/0/1
```

The following is sample output from the **show mka statistics interface** *interface-name* command:

```
Device# show mka statistics interface GigabitEthernet 1/0/1


MKA Statistics for Session
==========================
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.......... 1
   SAKs Rekeyed............ 0
   SAKs Received........... 0
   SAK Responses Received.. 1

MKPDU Statistics
   MKPDUs Validated & Rx... 89585
      "Distributed SAK".. 0
      "Distributed CAK".. 0
   MKPDUs Transmitted...... 89596
      "Distributed SAK".. 1
      "Distributed CAK".. 0
```

The following is sample output from the **show mka summary** command:

```
Device# show mka summary

Total MKA Sessions....... 1
      Secured Sessions... 1
      Pending Sessions... 0

================================================================================
Interface      Local-TxSCI         Policy-Name      Inherited
```

```
Key-Server
Port-ID        Peer-RxSCI          MACsec-Peers     Status             CKN
=============================================================================
Gi1/0/1        204c.9e85.ede4/002b p2               NO                 YES
43             c800.8459.e764/002a 1                 Secured
0100000000000000000000000000000000000000000000000000000000000000


MKA Global Statistics
=====================
MKA Session Totals
   Secured..................... 1
   Reauthentication Attempts.. 0

   Deleted (Secured).......... 0
   Keepalive Timeouts......... 0

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

SA Statistics
   SAKs Generated............. 1
   SAKs Rekeyed............... 0
   SAKs Received.............. 0
   SAK Responses Received..... 1

MKPDU Statistics
   MKPDUs Validated & Rx...... 89589
      "Distributed SAK"..... 0
      "Distributed CAK"..... 0
   MKPDUs Transmitted......... 89600
      "Distributed SAK"..... 1
      "Distributed CAK"..... 0

MKA Error Counter Totals
========================
Session Failures
   Bring-up Failures.............. 0
   Reauthentication Failures........ 0
   Duplicate Auth-Mgr Handle........ 0

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

```
CA Failures
   Group CAK Generation..... 0
   Group CAK Encryption/Wrap..... 0
   Group CAK Decryption/Unwrap..... 0
   Pairwise CAK Derivation..... 0
   CKN Derivation..... 0
   ICK Derivation..... 0
   KEK Derivation..... 0
   Invalid Peer MACsec Capability... 0
MACsec Failures
   Rx SC Creation..... 0
   Tx SC Creation..... 0
   Rx SA Installation..... 0
   Tx SA Installation..... 0

MKPDU Failures
   MKPDU Tx..... 0
   MKPDU Rx Validation..... 0
   MKPDU Rx Bad Peer MN..... 0
   MKPDU Rx Non-recent Peerlist MN.. 0
```

Additional References for MACsec Encryption

Standards and RFCs

Standard/RFC	Title
IEEE 802.1AE-2006	*Media Access Control (MAC) Security*
IEEE 802.1X-2010	*Port-Based Network Access Control*
IEEE 802.1AEbw-2013	*Media Access Control (MAC) Security (Amendment to IEEE 802.1AE-2006)—Extended Packet Numbering (XPN)*
IEEE 802.1Xbx-2014	*Port-Based Network Access Control (Amendment to IEEE 802.1X-2010)*
RFC 4493	*The AES-CMAC Algorithm*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	Support & Downloads page on Cisco.com

Feature History for MACsec Encryption

This table provides release and related information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature	Feature Information
Cisco IOS XE 17.13.1	Certificate-based MACsec encryption	Support for this feature was introduced for the Cisco Catalyst ESS9300 Embedded Series Switch in this release.
Cisco IOS XE Cupertino 17.8.x	MACsec encryption	MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. Support for this feature was introduced for Cisco Catalyst IE9300 Rugged Series Switches in this release.

Use the Cisco Feature Navigator to find information about platform and software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn.

Overview of PQC for MACsec with EAP-TLS

Post Quantum Cryptography (PQC) describes cryptographic algorithms that remain secure against attacks from quantum computers. Quantum computers can break many classical encryption methods. Widely used encryption methods—such as Diffie-Hellman (DH) and Elliptic Curve Diffie-Hellman (ECDH)—are vulnerable as quantum computing technology advances.

Implementing quantum-safe encryption now is critical for the long-term confidentiality and integrity of sensitive communications, particularly for government, military, and financial sectors.

PQC for MACsec with EAP-TLS

MACsec encryption with EAP-TLS authentication establishes secure communication between switches by using certificate-based mutual authentication to derive keys for MACsec encryption. This process ensures that data traffic on the MACsec link is encrypted and decrypted and maintains integrity and confidentiality throughout the session.

Post Quantum Cryptography for MACsec with EAP-TLS provides line-rate Layer 2 encryption with quantum-safe mutual authentication. This section provides the configuration information for MACsec with EAP-TLS sessions.

Key elements of PQC

PQC is implemented by using the algorithms available as part of Module-lattice based Cryptography. Module-lattice based Cryptography is a quantum-resistant public-key cryptography that uses structured, high-dimensional grid points called lattices to secure data. This approach combines module-lattice based Cryptography (ML-KEM) with an elliptic curve function such as p256.

This combination provides enhanced security by protecting against both classical and quantum attacks while maintaining compatibility and performance with existing cryptographic protocols.

This table lists the supported algorithms and the areas in which each algorithm can be used.

Table 2: Key elements of PQC

Algorithm	What this algorithm means	Security level
• mlkem512 • mlkem768 • mlkem1024	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks.	The security level of this implementation is balanced and is best suited for deployments where performance metrics are critical.
• p256_mlkem512 • p384_mlkem768 • p521_mlkem1024	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. p256, p384, and p521 refer to the traditional (classical) component. The algorithm uses Elliptic Curve Diffie-Hellman (ECDH) with the NIST P-256, P-384, or P-521 curves to maintain security against current non-quantum threats.	The security level of this implementation is the highest and is best suited for deployments where maximum security is needed.

Algorithm	What this algorithm means	Security level
• x25519_mlkem512 • x448_mlkem768 • X25519MLKEM76	ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks. X448 and X25519 are widely used classical elliptic curve Diffie-Hellman key exchange algorithms. This hybrid PQC key exchange algorithm is used to secure key exchanges against quantum and classical threats in Cisco's PQC-enabled network protocols.	The security level of this implementation is the highest and is best suited for deployments where maximum security is needed.
• SecP256r1MLKEM768 • SecP384r1MLKEM1024	secp384r1 and secp521r1 are standard, classical Elliptic Curve Diffie-Hellman (ECDH) groups used for key exchange during the TLS 1.3 handshake. ML-KEM is a PQC algorithm based on module-lattice cryptography, providing resistance against quantum attacks.	The security level of this implementation is good and is best suited for most deployments.
• secp256r1 • secp384r1 • secp521r1	Secp256r1, secp384r1, and secp521r1 are standard, classical ECDH groups used for key exchange during the TLS 1.3 handshake. These are the standard, non-quantum-resistant cryptographic curves that the system uses to ensure interoperability and maintain security when PQC is not required or not supported by the peer.	If a PQC-compliant handshake cannot be established (for example, when the pqc-type is set to all or non-pqc), the system falls back to using these classic groups. This approach is best suited for deployments where PQC is not chosen.

PQC implementation in EAP-TLS

PQC in Extensible Authentication Protocol Transport Layer Security (EAP-TLS) is implemented through negotiated key-exchange configuration. TLS 1.3 is mandatory for this implementation. This implementation secures Ethernet traffic through MACsec encryption. It combines IEEE 802.1X port-based authentication and EAP-TLS certificates for mutual authentication and automated key derivation.

MACsec encryption with EAP-TLS secures switch-to-switch communication. Certificate-based mutual authentication derives cryptographic keys, which are managed by the MACsec Key Agreement (MKA) protocol to ensure data confidentiality and integrity across Ethernet interfaces.

MACsec encryption uses EAP-TLS authentication to establish secure communication between switches. Certificate-based mutual authentication is leveraged to derive keys for MACsec encryption.

Summary

The key components involved in the process are

- Switches (authenticator/supplicant): Systems that perform MACsec encryption and participate in 802.1X authentication, acting as either the authenticator (facilitates authentication) or the supplicant (seeks authentication).
- Authentication server: An entity that provides authentication services to an authenticator, verifying supplicant credentials and facilitating EAP-TLS communication.
- Certificate Authority (CA) server: Issues and manages digital certificates used for mutual authentication in EAP-TLS.
- EAP-TLS (Extensible Authentication Protocol-Transport Layer Security): The authentication method used for mutual authentication between the authentication server and the client (supplicant) using certificates.
- Master Session Key (MSK): A cryptographic key generated upon successful EAP-TLS authentication.
- Connectivity Association Key (CAK): Derived from the MSK, this key is used by the MACsec Key Agreement (MKA) protocol. CAK is used to derive Secure Association Key (SAK), which performs the actual encryption and authentication of data packets on the link.
- Connectivity Association Key Name (CKN): Derived from the EAP session ID. This name identifies the CAK.

Workflow

These stages describe how MACsec encryption using EAP-TLS authentication works:

1. Initiation: A supplicant switch initiates 802.1X port-based authentication on a physical Ethernet interface with an authenticator switch.
2. Mutual authentication (EAP-TLS): The authentication server and the supplicant switch perform mutual authentication using digital certificates via the EAP-TLS method. This requires both devices to have valid certificates issued by a trusted Certificate Authority.
3. Master session key generation: Upon successful EAP-TLS authentication, a Master Session Key (MSK) is generated.
4. Key derivation: The MSK is then used to derive the Connectivity Association Key (CAK), and the Connectivity Association Key Name (CKN) is derived from the EAP session ID.
5. MACsec Key Agreement (MKA): The derived CAK and CKN are utilized by the MKA protocol to establish and maintain secure MACsec encryption between the switches on the interface.

Result

This process enables robust MACsec encryption between two switches, ensuring data confidentiality and integrity on Ethernet interfaces through secure, certificate-based authentication and automated key management.

Prerequisites

This list specifies the prerequisites before you configure PQC for MACSec with EAP-TLS.

- Ensure MACsec is enabled on the authenticator's (the switch that facilitates authentication) interface and ensure that the supplicant (the switch that seeks authentication) is capable of MACsec.

Restrictions

This list provides the limitations for configuring PQC in MACsec with EAP-TLS.

- In Cisco IOS XE 26.1.1, PQC for EAP-TLS is not supported in Federal Information Processing Standards (FIPS) mode.
- PQC for EAP-TLS is not supported with TLS versions 1.0 and 1.2.
- Certificate Revocation List (CRL) revocation is not supported for MACsec with EAP-TLS.
- PQC is supported with only local authentication for MACsec EAP -TLS.

Configuration matrix for MACsec with EAP-TLS

The protocol and TLS version matrix lists the combinations required for PQC to operate as expected for EAP-TLS. In this matrix, both device 1 and device 2 are switches.

Device 1 protocol and version	Device 2 protocol and version	Negotiated TLS version	Authentication status
EAP-TLS – 1.3	EAP-TLS – 1.3	1.3	Authentication successful
EAP-TLS – all	EAP-TLS – all	1.3	Authentication successful
EAP-TLS-1.2	EAP-TLS-all	1.2	Authentication successful
EAP-TLS – 1.3	EAP-TLS – 1.2		Authentication failure
EAP-TLS – 1.0	EAP-TLS – 1.3		Authentication failure

Supported key-exchange type for the algorithms

The table specifies the key-exchange type supported for each algorithm. The table lists these algorithms used to verify whether PQC is supported on your platform.

- **All**: all PQC, non-PQC and hybrid algorithms are supported.
- **Hybrid**: combined PQC and non-PQC algorithms are supported.

- **Non-PQC**: these are the classic cryptographic algorithms without PQC.
- **PQC**: PQC algorithms.

Version or key-exchange-type	Classic	PQC	Hybrid	All
1.3	Supported	Supported	Supported	Supported
1.2	Supported	Not supported	Not supported	Supported
1.0	Supported	Not supported	Not supported	Supported

Workflow to set up PQC for MACsec with EAP-TLS

This section describes how an authenticator and a supplicant can complete EAP-TLS authentication using ML-KEM for the TLS key exchange. This process results in the establishment of a secure MACsec session.

The steps to set up this configuration are:

1. Enable EAP-TLS with support for PQC ML-KEM on the supplicant as well as the authenticator.
2. Install and trust valid client and server certificates on the respective devices.
3. Connect the supplicant to the authenticator's MACsec-enabled port.
4. Initiate the authentication process by sending traffic from the supplicant or enabling the interface.
5. Verify PQC for EAP-TLS sessions.

Configure EAP-TLS 1.3

Follow these steps to configure TLS 1.3. If the peer device supports TLS version 1.3, authentication succeeds through EAP-TLS 1.3; otherwise, authentication fails.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **access-session tls-version** *version* command to configures the TLS 1.3 version.

Example:

```
Switch (config)# access-session tls-version <1.3>
```

Sample configuration

```
Switch (config)# access-session tls-version ?
  1.0  Version 1.0
  1.2  Version 1.2
  1.3  Version 1.3
  all  All TLS version
```

By default, **all** is configured on a device. When both the devices support TLS 1.3, TLS 1.3 negotiation is successful. Else, an older version of TLS version is enabled. In this scenario, PQC does not work.

Configure PQC for EAP-TLS sessions

Follow these steps to enable PQC for EAP-TLS sessions.

Procedure

Step 1 Use the **configure terminal** command to enter global configuration mode.

Example:

```
Switch# configure terminal
```

Step 2 Use the **access-session pqc-type** *pqc* to configure the PQC type.

Example:

```
Switch (config)# access-session pqc-type pqc
```

Note
If a device does not support PQC, the system displays this warning message:

```
Switch(config)# access-session pqc-type pqc
Warning! This platform only supports non-pqc method
```

Sample configuration

```
Switch(config)# access-session pqc-type ?
  all        All pqc, non-pqc and hybrid algorithms will be supported
  hybrid     Combined PQC and NON-PQC algorithms (Key dervived with combination of both)
  non-pqc    Classic cryptographic algorithms
  pqc        Post-Quantum Cryptographic algorithms
```

Verify PQC for EAP-TLS sessions

Use these commands to verify whether PQC implementation is successful for EAP-TLS sessions.

Command	Purpose	What to look out for
Device(config)# access-session tls-version ?	Verify the TLS version	The TLS version should read 1.3 for this functionality to work.
show running-config all | i pqc-type	Verify the configured PQC type	
show dot1x interface <intf> detail	Verify the EAP profile is set up and the EAP method	The EAP method should read TLS.