MACsec Encryption

Feature history for MACsec encryption

This table provides release and platform support 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 name and description	Supported platform
Cisco IOS XE 26.1.1	MACsec encryption: MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. Support for switch-to-switch MACsec encryption was introduced.	Cisco C9350 Series Smart Switches

Understand MACsec encryption

MACsec is the IEEE 802.1AE standard for authenticating and encrypting packets between two MACsec-capable devices. The switch supports MACsec encryption for switch-to-switch security using MKA-based key exchange. MKA supports switch-to-switch links.

Note Enable switch-to-switch MACSec to encrypt all traffic, while EAP-over-LAN (EAPOL) packets remain unencrypted.

Link layer security includes packet authentication between switches. It also provides optional MACsec encryption.

Table 1: MACsec support on switch ports

Connections	MACsec support
Switch-to-switch	MACsec MKA encryption

MACsec key agreement

The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys. MACsec, defined in 802.1AE, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. MKA and MACsec are implemented after successful authentication using the 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). The switch decrypts frames from the MKA peer and calculates the correct ICV 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. 802.1x-REV defines the basic requirements of MKA. The MKA Protocol extends 802.1x by allowing peer discovery with confirmed mutual authentication and by facilitating the sharing of MACsec secret keys to protect data exchanged by the peers.

MKA EAP authentication

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. Enter the EAP session ID to generate a secure connectivity association key name (CKN). The switch authenticates switch-to-switch.

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 removed after the MKA lifetime of 6 seconds if no MKPDU is 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.

MKA policies

Apply a defined MKA policy to an interface to enable MKA. Remove the MKA policy to disable MKA on that interface. You can configure these options:

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

Note The confidentiality offset of 50 bytes is not supported when using MACsec XPN Cipher Suites.

MKA statistics

The system aggregates some MKA counters globally, while it updates others both globally and per session. You can also obtain information about the status of MKA sessions. See "Example: Display MKA information" for further information.

MACsec features

This section describes the features supported by MACsec.

Key lifetime and hitless key rollover

A MACsec key chain can have multiple pre-shared keys (PSK). Each key is configured with a key ID and an optional lifetime that specifies when 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. You can configure the key's time zone to be local or use UTC, with the default being UTC.

You can achieve a key rollover to the next key within the same key chain by configuring a second key and setting 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, synchronize MACsec key chains using Network Time Protocol (NTP), ensuring all devices use the same time zone. If all participating devices are not synchronized, rekeying the connectivity association key (CAK) will not occur simultaneously across all devices.

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

Fallback key

The Fallback Key feature establishes an MKA session with the pre-shared fallback key when the primary pre-shared key (PSK) cannot establish a session due to a key mismatch. This feature prevents downtime and ensures a seamless traffic transition during CAK mismatch. The fallback key chain serves as a last resort key. The fallback key feature applies only to PSK-based MKA or MACsec sessions.

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 can be reordered when transmitted through a Metro Ethernet service provider network because of prioritization and load balancing.

A replay window supports 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 of 0 enforces strict

reception ordering. Configure the replay window size in the range of 0 to 2^{32}- 1. In case of XPN cipher suite, maximum replay window size is 2^{30}- 1, and if a higher window size is configured, the window size gets restricted to 2^{30}- 1. If the cipher suite changes to a non-XPN cipher suite, there is no restriction; the configured window size is used.

MACsec must secure policy

With must-secure mode enabled on switch-to-switch connections, EAPOL traffic is unencrypted and all other traffic is encrypted. All unencrypted packets are dropped. Must-secure support is enabled on both the ingress and the egress, and it is supported for MKA.

Must-secure mode is enabled by default.

MACsec extended packet numbering

Every MACsec frame contains a unique 32-bit packet number (PN) for a given Security Association Key (SAK). With non-XPN cipher suites, upon PN exhaustion, that is, after reaching 75% of 2^{32}- 1), SAK rekey takes place to refresh the data plane keys. For high-capacity links like 40 Gb/s, the PN exhausts within a few seconds, requiring frequent SAK rekeys to the control plane. The XPN cipher suites eliminate the need for frequent SAK rekey that may occur in high capacity links. XPN supports the protection of up to 2^{64} frames by a single SAK. Rekey occurs when 75% of 2^{64}- 1 frames are reached, which takes several years to exhaust. This ensures that frequent SAK rekeys are not needed on high-speed links. XPN is a mandatory requirement for FIPS/CC compliance on high-speed links such as 40 Gb/s, 100 Gb/s, and so on.

MACsec SAK rekey

These SAK rekey options are supported:

- Volume-based Rekey: The MACsec frame uses a 32-bit packet number (PN). Non-XPN cipher suites, GCM-AES-128, and GCM-AES-256 protect up to 2^{32} frames with a single SAK. Rekey is triggered after reaching 75% of 2^{32}- 1 frames. XPN cipher suites, GCM-AES-XPN-128, or GCM-AES-XPN-256 allows upto 2^{64} frames to be protected with a single SAK without changing the MACsec frame structure. The MACsec frame includes only the lowest 32 bits while both the sending and receiving ends maintain the most significant 32 bits. The most significant 32 bits of the PN is incremented at the receiving end when the most significant bits (MSB) of lowest acceptable packet number (LAPN) is set, and the MSB of the PN value in the received MACsec frame is 0.
- Time-based Rekey: You can manually set the SAK rekey using a timer-based rekey, allowing you to start rekeying the SAK at specified intervals. Use the **sak-rekey interval** *time-interval* command in MKA policy configuration mode to set the SAK rekey interval. This MKA policy is then applied to the interface.

Note Volume-based rekey overrides time-based rekey.

MACsec for port channel

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

Note Etherchannel links, formed as part of the port channel, can be congruent or disparate. This means the links can be either 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.

It is recommended that you enable MKA MACsec on all the member ports for better security of the port channel.

Certificate-based MACsec encryption

Configure MACsec MKA between device switch-to-switch ports using certificate-based MACsec encryption. Certificate-based MACsec encryption 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 encryption, for authentication to the AAA server.

MACsec PQC

Post-Quantum Cryptography (PQC) is designed to provide cryptographic protection that can withstand attacks from quantum computers, addressing vulnerabilities in traditional public-key algorithms like RSA and classical Elliptic Curve Diffie-Hellman (ECDH), which are susceptible to quantum-based cryptanalysis.

PQC implementations use module-lattice–based cryptography which relies on structured and high-dimensional lattice mathematics. These mathematical problems are currently considered resistant to both classical and quantum attacks.

PQC is supported for MACsec key establishment using Module-Lattice Key Encapsulation Mechanism (ML-KEM) within the TLS 1.3 handshake. ML-KEM provides quantum-resistant key exchange capabilities.

Note

- PQC with certificate-based MACsec encryption is supported for local authentication only.
- PQC and hybrid algorithms do not support FIPS mode. FIPS mode is only supported for non-PQC algorithms.

Implement MACsec PQC

PQC is implemented as a post-quantum key exchange model for MACsec, leveraging quantum-resistant algorithms.

Summary

When PQC is enabled:

1. TLS 1.3 is used for MACsec authentication and key establishment.
2. ML-KEM is negotiated as the key encapsulation mechanism.
3. Session keys are derived using ML-KEM only when PQC is enforced, ensuring quantum-resistant session key establishment.
4. Hybrid key exchange modes combining classical and PQC algorithms are supported using different modes.
5. Classical elliptic curve fallback is not used when PQC is enforced.

This ensures that MACsec session keys are established using a quantum-resistant mechanism, protecting encrypted traffic between peer devices from future quantum-based threats.

Workflow

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

1. Initiation: A supplicant device initiates 802.1X port-based authentication on a physical Ethernet interface with an authenticator device.
2. EAP message exchange: The authenticator device forwards EAP messages between the supplicant and the configured external authentication server (e.g., RADIUS) using EAP as the transport.
3. Mutual authentication (EAP-TLS): The authentication server and the supplicant device 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.
4. Master session key generation: Upon successful EAP-TLS authentication, a Master Session Key (MSK) is generated.
5. Key derivation: The MSK is then used to derive the CAK, and the CKN is derived from the EAP session ID.
6. MACsec Key Agreement (MKA): The derived CAK and CKN are utilized by the MKA protocol to establish and maintain secure MACsec encryption between the devices on the interface.

Result

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

Configuration matrix for MACsec PQC

This table list the protocol and TLS version combinations required for PQC to operate as expected for EAP-TLS.

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.

- **All**: PQC, non-PQC, and hybrid algorithms
- **Hybrid**: PQC and non-PQC algorithms
- **Non-PQC**: classic cryptographic algorithms without PQC

- **PQC**: PQC algorithms

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

Recommendations for MACsec encryption

This section list the recommendations for configuring MACsec encryption:

- Use Bidirectional Forwarding and Detection (BFD) timer value as 750 milliseconds for 10Gbps ports and 1.25 seconds for any port with speed above 10Gbps.
- 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.
- Use MACsec MKA encryption.
- Do not use the **delay-protection** command when defining MKA policy if MACsec scale sessions are configured.

Prerequisites for MACsec encryption

Prerequisites for MACsec encryption

Here are the 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.
- Configure 802.1x authentication and AAA on your device.
- Configure the **flowcontrol receive desired** command on all MACsec-enabled ports to enable flowcontrol explicitly.

Prerequisites for certificate-based MACsec

Here are the prerequisites for certificate-based MACsec:

- Ensure that you have a Certificate Authority (CA) server configured for your network.
- Generate a CA certificate.
- Configure Cisco Identity Services Engine (ISE) Release 2.0.

- Synchronize participating devices, the CA server, and Cisco Identity Services Engine (ISE) using Network Time Protocol (NTP). Synchronize time on all your devices to ensure certificates are validated.

Restrictions for MACsec encryption

- MACsec configuration can be applied and removed only on the individual member ports of an EtherChannel. MACsec configuration is not supported directly on EtherChannel ports. To remove MACsec configuration, you must first unbundle the member ports from the EtherChannel, and then remove it from the individual member ports.
- MACsec with MKA is supported only on point-to-point links.
- GCM-AES-256 and XPN cipher suites (GCM-AES-XPN-128 and GCM-AES-XPN-256) are supported only with Network Advantage license.
- The MACsec Cipher announcement is not supported for MACsec Extended Packet Numbering (XPN) Ciphers and switch-to-switch MACsec connections.
- MACsec XPN Cipher Suites are not supported in switch-to-host MACsec connections.
- MACsec XPN Cipher Suites do not provide confidentiality protection with a confidentiality offset. These features are not supported together in switch-to-switch MACsec connections.
- As per IEEE standards, the maximum value of replay window is 2^{30-1} for MACsec XPN Cipher Suites. Even if you configure a higher value than this, the system restricts it to 2^{30-1} only.
- Certificate-based MACsec is supported only if the access-session is configured as closed or in multiple-host mode. Other configuration modes are not supported.
- If the **dot1q tag vlan native** command is configured globally, the dot1x reauthentication will fail on trunk ports.
- MACsec is not supported with Multicast VPN (mVPN).
- PSK fallback key chain is not supported on Ethernet Virtual Circuit (EVC) and point-to-multipoint cases.
- PSK fallback key chain supports an 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.
- EAPOL packets of EtherType 0x888E are not intercepted by the interface if MACsec or dot1x is not enabled on the interface.
- P2P protocols such as STP or CDP will not function if intermediate switches are present between two MACsec endpoints.
- Network-Based Application Recognition (NBAR) is not supported on MACsec switch-to-host connections.
- EAPOL bypass and custom EAPOL (876F) are not supported.
- The **should-secure** access mode is not supported.
- GRE/VxLAN with MACsec is not supported.
- ClearTag (WAN) MACsec is not supported.

Configure MACsec encryption

These sections provide information about the various tasks that comprise MACsec encryption.

Configure MKA and MACsec

By default, MACsec is disabled. No MKA policies are configured.

Configure an MKA policy

Perform these steps to create an MKA Protocol policy. Note that MKA also requires that you enable 802.1x.

Procedure

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 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 4 **key-server** *priority*

Example:

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

Configures MKA key server options and set 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.

Step 5 **include-icv-indicator**

Example:

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

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

Step 6 **macsec-cipher-suite** {*gcm-aes-128* | *gcm-aes-256*}

Example:

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

Configures a cipher suite for deriving SAK with 128-bit or 256-bit encryption.

Step 7 **confidentiality-offset** *offset-value*

Example:

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

Set the confidentiality (encryption) offset for each physical interface.

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

Step 8 **ssci-based-on-sci**

Example:

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

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

Step 9 **end**

Example:

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

Exit enters MKA policy configuration mode and returns to privileged EXEC mode.

Step 10 **show mka policy**

Example:

```
Device# show mka policy
```

Displays MKA policy configuration information.

Configure MKA MACsec using PSK

These sections provide information about the various tasks to configure MKA MACsec using PSK.

Configure MACsec MKA using PSK

Perform these steps to configure MACsec MKA policies using a Pre Shared Key (PSK).

Procedure

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

Enter your password, if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

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

Perform these steps to configure MACsec MKA policies on an interface using a Pre Shared Key (PSK).

Procedure

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** *type number*

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

We do not recommend that you 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.

Configure certificate-based MACsec encryption

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 encryption Profiles and IEEE 802.1x Credentials
- Configure MKA MACsec using certificate-based MACsec encryption on Interfaces

Generate key pairs

Procedure

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 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.

Procedure

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.

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. This 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 this task to set up manual certificate enrollment:

Procedure

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*

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
```

Specifies that serial numbers 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 **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.

Step 14 **show crypto pki certificate** *trustpoint name*

Example:

```
Device# show crypto pki certificate  ka
```

Displays information about the certificate for the trust point.

Configure certificate based MACSec encryption

Perform this task to apply MACsec MKA using certificate-based MACsec encryption to interfaces.

Procedure

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** *type number*

Example:

```
Device(config)# interface gigabitethernet 0/2/1
```

Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.

Step 4 **macsec network-link**

Example:

```
Device(config-if)# macsec network-link
```

Enables MACsec on the interface.

Step 5 **authentication periodic**

Example:

```
Device(config-if)# authentication periodic
```

Enables reauthentication for this port.

Step 6 **authentication timer reauthenticate interval**

Example:

```
Device(config-if)# authentication timer reauthenticate interval
```

Sets the reauthentication interval.

Step 7 **access-session host-mode multi-host**

Example:

```
Device(config-if)# access-session host-mode multi-host
```

Allows hosts to gain access to the interface.

Step 8 **access-session closed**

Example:

```
Device(config-if)# access-session closed
```

Prevents preauthentication access on the interface.

Step 9 **access-session port-control auto**

Example:

```
Device(config-if)# access-session port-control auto
```

Sets the authorization state of a port.

Step 10 **dot1x pae both**

Example:

```
Device(config-if)# 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-if)# dot1x credentials profile
```

Assigns a 802.1x credentials profile to the interface.

Step 12 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Step 13 **show macsec interface** *interface-id*

Example:

```
Device# show macsec interface GigabitEthernet 1/0/1
```

Displays MACsec details for the interface.

Configure PQC ML-KEM for MACsec with EAP-TLS

Perform this task to configure PQC ML-KEM for MACsec with EAP-TLS.

Procedure

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 **eap profile** *profile-name*

Example:

```
Device(config)# eap profile EAP-PROFILE
```

Creates an EAP profile, and enters EAP profile configuration mode.

Step 4 **method tls**

Example:

```
Device(config-eap-profile)# method tls
```

Enables Transport Layer Security (TLS).

Step 5 **pki-trustpoint** *trustpoint-label*

Example:

```
Device(config-eap-profile)# pki-trustpoint tsp1
```

Specifies PKI trustpoints for use with the RSA signature authentication method.

Step 6 **exit**

Example:

```
Device(config-eap-profile)# exit
```

Exits EAP profile configuration mode, and returns to global configuration mode.

Step 7 **access-session tls-version all**

Example:

```
Device(config)# access-session tls-version all
```

Configures the device to allow all supported TLS versions for access sessions.

Step 8 **access-session pqc-type pqc**

Example:

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

Configures PQC algorithms for access sessions.

Step 9 **interface** *type number*

Example:

```
Device(config)# interface TenGigabitEthernet0/1/1
```

Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.

Step 10 **macsec**

Example:

```
Device(config-if)# macsec
```

Enables MACsec on the interface.

Step 11 **authentication periodic**

Example:

```
Device(config-if)# authentication periodic
```

Enables reauthentication for this port.

Step 12 **authentication timer reauthenticate interval**

Example:

```
Device(config-if)# authentication timer reauthenticate interval
```

Sets the reauthentication interval.

Step 13 **access-session host-mode multi-domain**

Example:

```
Device(config-if)# access-session host-mode multi-domain
```

Allows hosts to gain access to the interface.

Step 14 **access-session closed**

Example:

```
Device(config-if)# access-session closed
```

Prevents preauthentication access on the interface.

Step 15 **access-session port-control auto**

Example:

```
Device(config-if)# access-session port-control auto
```

Sets the authorization state of a port.

Step 16 **dot1x pae both**

Example:

```
Device(config-if)# dot1x pae both
```

Configures the port as an 802.1X port access entity (PAE) supplicant and authenticator.

Step 17 **dot1x credentials profile**

Example:

```
Device(config-if)# dot1x credentials profile
```

Assigns a 802.1x credentials profile to the interface.

Step 18 **dot1x authenticator eap profile** *profile-name*

Example:

```
Device(config-if)# dot1x authenticator eap profile EAP-PROFILE
```

Configures the interface to use EAP profile during 802.1x authentication

Step 19 **dot1x supplicant eap profile** *profile-name*

Example:

```
Device(config-if)# dot1x supplicant eap profile EAP-PROFILE
```

Assigns the EAP profile to the 802.1X supplicant on the interface.

Step 20 **service-policy type control subscriber** *control-policy name*

Example:

```
Device(config-if)# service-policy type control subscriber control-policy1
```

Applies the specified subscriber control policy to the interface.

Step 21 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Configure MACsec XPN

These sections provide information about the various tasks to configure MACsec XPN.

Configure an MKA Policy for XPN

Perform these steps to configure XPN in an MKA policy:

Procedure

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 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 4 **macsec-cipher-suite** { *gcm-aes-128* | *gcm-aes-256* | *gcm-aes-xpn-128* | *gcm-aes-xpn-256*}

Example:

```
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-xpn-256
```

Configures cipher suite for deriving SAK with 128-bit and 256-bit encryption for XPN.

Step 5 **sak-rekey interval** *time-interval*

Example:

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

(Optional) Configures the SAK rekey interval (in seconds). The range is from 30 to 65535. By default, the SAK rekey interval occurs automatically depending on the interface speed.

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

Step 6 **end**

Example:

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

Exits MKA policy configuration mode and returns to privileged EXEC mode.

Apply the XPN MKA policy to an interface

Perform this task to apply the XPN MKA policy to an interface.

Procedure

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-name*

Example:

```
Device(config)# interface FortyGigabitEthernet 1/0/1
```

Identifies the MACsec interface, and enters interface configuration mode. The interface must be a physical interface.

Step 4 **mka policy** *policy-name*

Example:

```
Device(config-if)# mka policy mka-xpn-policy
```

Applies the XPN MKA protocol policy to the interface.

Step 5 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Configure MKA MACsec for port channel

This topic describes about MKA MACsec for port channel.

Configure MKA MACsec for port channel using PSK

Perform these steps to configure MKA policies on an interface using a Pre-Shared Key (PSK).

Procedure

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.

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 either on the physical interface or the subinterface but 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 associated with this channel group is automatically created if the port channel does not already exist. For mode, select one of these 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 port group in the **on** mode is connected to another port group in the **on** mode.
- **active**: Enables LACP only if an 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)# end
```

Exits interface configuration mode and returns to privileged EXEC mode.

Configure port channel logical interfaces for Layer 2 EtherChannels

Perform this task to create a port channel interface for a Layer 2 EtherChannel.

Procedure

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, and enters interface configuration mode.

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.

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

Perform this task to create a port channel interface for a Layer 3 EtherChannel.

Procedure

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.

Configuration examples

These sections provide configuration examples for MACsec encryption.

Example: Configure 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 switch-to-host 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)# authentication event linksec fail action authorize vlan 1
Device(config-if)# authentication host-mode multi-domain
Device(config-if)# authentication linksec policy must-secure
Device(config-if)# authentication 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
```

Example: Configure 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

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
```

This 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 1100000000000000000000000000000000000000000000000000000000000000
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
```

Example: Configure switch-to-switch MACsec MKA using certificate-based MACsec encryption

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

```
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)# dot1x authenticator eap profile profile_eap_tls
Device(config-if)# end
```

Example: Configure MACsec XPN

This example shows how to configure MACsec MKA XPN policy:

```
Device> enable
Device# configure terminal
Device(config)# mka policy mka-xpn-policy
Device(config-mka-policy)# macsec-cipher-suite gcm-aes-xpn-256
Device(config-mka-policy)# end
```

This example shows how to apply MACsec MKA XPN policy to an interface:

```
Device> enable
Device# configure terminal
Device(config)#interface Fo 1/0/1
Device(config-if)# mka policy mka-xpn-policy
Device(config-if)# end
```

This is a sample output of the **show mka sessions details** command with 128-bit XPN Cipher Suite configured:

```
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)........... 01000000000000000000000000000000000000000000000000000000000000000000
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......... 0080C20001000003 (GCM-AES-XPN-128)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

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

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
  ------------------------------------------------------------------------
```

This is a sample output of the **show mka sessions details** command with 256-bit XPN Cipher Suite configured:

```
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)........... 01000000000000000000000000000000000000000000000000000000000000000000
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......... 0080C20001000004 (GCM-AES-XPN-256)
MACsec Capability........ 3 (MACsec Integrity, Confidentiality, & Offset)
MACsec Desired........... YES

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

Example: Configure MACsec MKA for port channel using PSK

Etherchannel mode: Static/On

This is a 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
```

This is 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

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
```

This is 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

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
2      Po2(RU)          -        Te1/0/1(P)  Te1/0/2(P)
```

Example: Display MKA information

This is a 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
01000000000000000000000000000000000000000000000000000000000000000000
```

This is a 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
```

This 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
  ------------------------------------------------------------------------
```

This is a 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)........... 01000000000000000000000000000000000000000000000000000000000000000000000000000000
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
  ------------------------------------------------------------------------
```

This is a 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
```

This is a 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
```

This is a 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
```

This is a 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
```

This is a 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
01000000000000000000000000000000000000000000000000000000000000000000



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
```

This is a sample output from the **show macsec interface** command:

```
Device# show macsec interface HundredGigE 2/0/4

 MACsec is enabled
  Replay protect : enabled
  Replay window : 0
  Include SCI : yes
  Use ES Enable : no
  Use SCB Enable : no
  Admin Pt2Pt MAC : forceTrue(1)
  Pt2Pt MAC Operational : no
  Cipher : GCM-AES-128
  Confidentiality Offset : 0

 Capabilities
  ICV length : 16
  Data length change supported: yes
  Max. Rx SA : 16
  Max. Tx SA : 16
  Max. Rx SC : 8
  Max. Tx SC : 8
  Validate Frames : strict
  PN threshold notification support : Yes
  Ciphers supported : GCM-AES-128
                      GCM-AES-256
                      GCM-AES-XPN-128
                      GCM-AES-XPN-256

 Access control : must secure

 Transmit Secure Channels
  SCI : 3C5731BBB5850475
  SC state : inUse(1)
   Elapsed time : 7w0d
   Start time : 7w0d
   Current AN: 0
   Previous AN: -
   Next PN: 149757
   SA State: inUse(1)
   Confidentiality : yes
   SAK Unchanged : yes
   SA Create time : 00:04:41
   SA Start time : 7w0d
   SC Statistics
```

```
    Auth-only Pkts : 0
    Auth-only Bytes : 0
    Encrypted Pkts : 0
    Encrypted Bytes : 0
   SA Statistics
    Auth-only Pkts : 0
    Auth-only Bytes : 0
    Encrypted Pkts : 149756
    Encrypted Bytes : 16595088

  Port Statistics
   Egress untag pkts  0
   Egress long pkts  0

 Receive Secure Channels
  SCI : 3C5731BBB5C504DF
  SC state : inUse(1)
   Elapsed time : 7w0d
   Start time : 7w0d
   Current AN: 0
   Previous AN: -
   Next PN: 149786
   RX SA Count: 0
   SA State: inUse(1)
   SAK Unchanged : yes
   SA Create time : 00:04:39
   SA Start time : 7w0d
   SC Statistics
    Notvalid pkts 0
    Invalid pkts 0
    Valid pkts 0
    Late pkts 0
    Uncheck pkts 0
    Delay pkts 0
    UnusedSA pkts 0
    NousingSA pkts 0
    Validated Bytes 0
    Decrypted Bytes 0
   SA Statistics
    Notvalid pkts 0
    Invalid pkts 0
    Valid pkts 149784
    Late pkts 0
    Uncheck pkts 0
    Delay pkts 0
    UnusedSA pkts 0
    NousingSA pkts 0
    Validated Bytes 0
    Decrypted Bytes 16654544

  Port Statistics
   Ingress untag pkts  0
   Ingress notag pkts  631726
   Ingress badtag pkts  0
   Ingress unknownSCI pkts  0
   Ingress noSCI pkts  0
   Ingress overrun pkts  0
```

This is a sample output from the **show dot1x interface** *interface-name* **detail** command:

```
Device# show dot1x interface TenGigabitEthernet0/1/1 detail

Dot1x Info for TenGigabitEthernet0/1/1
PAE                     = BOTH
QuietPeriod             = 60
```

```
ServerTimeout             = 0
SuppTimeout               = 30
ReAuthMax                 = 2
MaxReq                    = 2
TxPeriod                  = 30
EAP profile               = EAP-PROFILE
Dot1x Info for
PAE                       = SUPPLICANT
StartPeriod               = 30
AuthPeriod                = 30
HeldPeriod                = 60
MaxStart                  = 3
Credentials profile       = DOT1X-CREDS
Dot1x Authenticator Client List
EAP Method                = TLS
Supplicant                = 78bc.1a60.d28a
Session ID                = 000000000000000C11F60339
Auth SM State         = AUTHENTICATED
Auth BEND SM State    = IDLE
Dot1x Supplicant Client List
```