Certification Authority Interoperability

This chapter describes how to configure certification authority (CA) interoperability, which is provided in support of the IPSec protocol. CA interoperability permits Cisco IOS devices and CAs to communicate so that your Cisco IOS device can obtain and use digital certificates from the CA. Although IPSec can be implemented in your network without the use of a CA, using a CA provides manageability and scalability for IPSec.

Prerequisites For Certification Authority

You need to have a certification authority (CA) available to your network before you configure this interoperability feature. The CA must support the Public Key Infrastructure (PKI) protocol, and the Simple Certificate Enrollment Protocol (SCEP) .

Restrictions for Certification Authority

When configuring your CA, the following restrictions apply:

- This feature should be configured only when you also configure both IPsec and Internet Key Exchange (IKE) in your network.
- The Cisco IOS software does not support CA server public keys greater than 2048 bits.

Information About Certification Authority

CA Supported Standards

Without certification authority (CA) interoperability, Cisco IOS devices could not use CAs when deploying IPSec. CAs provide a manageable, scalable solution for IPSec networks.

Cisco supports the following standards with this feature:

- IPSec—IPSec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPSec provides these security services at the IP layer; it uses Internet Key Exchange to handle negotiation of protocols and algorithms based on local policy, and to generate the encryption and authentication keys to be used by IPSec. IPSec can be used to protect one or more data flows between a pair of hosts, between a pair of security gateways, or between a security gateway and a host.
- Internet Key Exchange (IKE)—A hybrid protocol that implements Oakley and Skeme key exchanges inside the Internet Security Association Key Management Protocol (ISAKMP) framework. Although IKE can be used with other protocols, its initial implementation is with the IPSec protocol. IKE provides authentication of the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations.
- Public-Key Cryptography Standard #7 (PKCS #7)—A standard from RSA Data Security, Inc., used to encrypt and sign certificate enrollment messages.
- Public-Key Cryptography Standard #10 (PKCS #10)—A standard syntax from RSA Data Security, Inc. for certificate requests.
- RSA Keys—RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA keys come in pairs: one public key and one private key.
- X.509v3 certificates—Certificate support that allows the IPSec-protected network to scale by providing the equivalent of a digital ID card to each device. When two devices wish to communicate, they exchange digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or to manually specify a shared key at each peer). These certificates are obtained from a CA. X.509 is part of the X.500 standard of the ITU.

Purpose of CAs

Certificate authorities (CAs) are responsible for managing certificate requests and issuing certificates to participating IPSec network devices. These services provide centralized key management for the participating devices.

CAs simplify the administration of IPSec network devices. You can use a CA with a network containing multiple IPSec-compliant devices such as routers.

Digital signatures, enabled by public key cryptography, provide a means of digitally authenticating devices and individual users. In public key cryptography, such as the RSA encryption system, each user has a key pair containing both a public and a private key. The keys act as complements, and anything encrypted with one of the keys can be decrypted with the other. In simple terms, a signature is formed when data is encrypted with a user's private key. The receiver verifies the signature by decrypting the message with the sender's public key. The fact that the message could be decrypted using the sender's public key indicates that the holder of the private key, the sender, must have created the message. This process relies on the receiver's having a copy of the sender's public key and knowing with a high degree of certainty that it really does belong to the sender and not to someone pretending to be the sender.

Digital certificates provide the link. A digital certificate contains information to identify a user or device, such as the name, serial number, company, department, or IP address. It also contains a copy of the entity's public key. The certificate is itself signed by a certification authority (CA), a third party that is explicitly trusted by the receiver to validate identities and to create digital certificates.

In order to validate the signature of the CA, the receiver must first know the CA's public key. Normally this process is handled out-of-band or through an operation done at installation. For instance, most web browsers are configured with the public keys of several CAs by default. The Internet Key Exchange (IKE), an essential component of IPSec, can use digital signatures to scalably authenticate peer devices before setting up security associations.

Without digital signatures, one must manually exchange either public keys or secrets between each pair of devices that use IPSec to protect communications between them. Without certificates, every new device added to the network requires a configuration change on every other device with which it communicates securely. With digital certificates, each device is enrolled with a certification authority. When two devices wish to communicate, they exchange certificates and digitally sign data to authenticate each other. When a new device is added to the network, one simply enrolls that device with a CA, and none of the other devices needs modification. When the new device attempts an IPSec connection, certificates are automatically exchanged and the device can be authenticated.

Implementing IPsec Without CAs

Without a CA, if you want to enable IPsec services (such as encryption) between two Cisco devices, you must first ensure that each device has the key of the other device (such as an RSA public key or a shared key). This requirement means that you must manually perform one of the following operations:

- At each device, enter the RSA public key of the other device.
- At each device, specify a shared key to be used by both device.

In the above illustration, each device uses the key of the other device to authenticate the identity of the other device; this authentication always occurs when IPsec traffic is exchanged between the two devices.

If you have multiple Cisco devices in a mesh topology and wish to exchange IPsec traffic passing among all of those devices, you must first configure shared keys or RSA public keys among all of those devices.

Every time a new device is added to the IPsec network, you must configure keys between the new device and each of the existing devices. (In Figure 34, four additional two-part key configurations would be required to add a single encrypting device to the network.)

Consequently, the more devices there are that require IPsec services, the more involved the key administration becomes. This approach does not scale well for larger, more complex encrypting networks.

Implementing IPsec With CAs

With a CA, you do not have to configure keys between all the encrypting devices. Instead, you individually enroll each participating device with the CA, requesting a certificate for the device. When this has been accomplished, each participating device can dynamically authenticate all the other participating devices. This process is illustrated in the illustration.

To add a new IPsec device to the network, you need only configure that new device to request a certificate from the CA, instead of making multiple key configurations with all the other existing IPsec devices.

Implementing IPsec with Multiple Root CAs

With multiple root CAs, you no longer have to enroll a device with the CA that issued a certificate to a peer. Instead, you configure a device with multiple CAs that it trusts. Thus, a device can use a configured CA (a trusted root) to verify certificates offered by a peer that were not issued by the same CA defined in the identity of the device.

Configuring multiple CAs allows two or more devices enrolled under different domains (different CAs) to verify the identity of each other when using IKE to set up IPsec tunnels.

Through Simple Certificate Enrollment Protocol (SCEP), each device is configured with a CA (the enrollment CA). The CA issues a certificate to the device that is signed with the private key of the CA. To verify the

certificates of peers in the same domain, the device is also configured with the root certificate of the enrollment CA.

To verify the certificate of a peer from a different domain, the root certificate of the enrollment CA in the domain of the peer must be configured securely in the device.

During Internet Key Exchange (IKE) phase one signature verification, the initiator will send the responder a list of its CA certificates. The responder should send the certificate issued by one of the CAs in the list. If the certificate is verified, the device saves the public key contained in the certificate on its public key ring.

With multiple root CAs, VPN users can establish trust in one domain and easily and securely distribute it to other domains. Thus, the required private communication channel between entities authenticated under different domains can occur.

How CA Certificates Are Used by IPsec Devices

When two IPsec devices want to exchange IPsec-protected traffic passing between them, they must first authenticate each other—otherwise, IPsec protection cannot occur. The authentication is done with IKE.

Without a CA, a device authenticates itself to the remote device using either RSA-encrypted nonces or preshared keys. Both methods require that keys must have been previously configured between the two devices.

With a CA, a device authenticates itself to the remote device by sending a certificate to the remote device and performing some public key cryptography. Each device must send its own unique certificate that was issued and validated by the CA. This process works because the certificate of each device encapsulates the public key of the device, each certificate is authenticated by the CA, and all participating devices recognize the CA as an authenticating authority. This scheme is called IKE with an RSA signature.

Your device can continue sending its own certificate for multiple IPsec sessions, and to multiple IPsec peers until the certificate expires. When its certificate expires, the device administrator must obtain a new one from the CA.

CAs can also revoke certificates for devices that will no longer participate in IPsec. Revoked certificates are not recognized as valid by other IPsec devices. Revoked certificates are listed in a certificate revocation list (CRL), which each peer may check before accepting a certificate from another peer.

Registration Authorities

Some CAs have a registration authority (RA) as part of their implementation. An RA is essentially a server that acts as a proxy for the CA so that CA functions can continue when the CA is offline.

Some of the configuration tasks described in this document differ slightly, depending on whether your CA supports an RA.

How to Configure Certification Authority

Managing NVRAM Memory Usage

Certificates and certificate revocation lists (CRLs) are used by your device when a CA is used. Normally certain certificates and all CRLs are stored locally in the NVRAM of the device, and each certificate and CRL uses a moderate amount of memory.

The following certificates are normally stored at your device:

- Certificate of your device
- Certificate of the CA
- Root certificates obtained from CA servers (all root certificates are saved in RAM after the device has been initialized)
- Two registration authority (RA) certificates (only if the CA supports an RA)

CRLs are normally stored at your device according to the following conditions:

- If your CA does not support an RA, only one CRL gets stored in the device.
- If your CA supports an RA, multiple CRLs can be stored in the device.

In some cases, storing these certificates and CRLs locally will not present any difficulty. In other cases, memory might become a problem—particularly if the CA supports an RA and a large number of CRLs have to be stored on the device. If the NVRAM is too small to store root certificates, only the fingerprint of the root certificate is saved.

To save NVRAM space, specify that certificates and CRLs should not be stored locally, but should be retrieved from the CA when needed. This alternative will save NVRAM space but could result in a slight performance impact. To specify that certificates and CRLs should not be stored locally on your device, but should be retrieved when required, enable query mode.

If you do not enable query mode now, you can do it later even if certificates and CRLs have are already stored on the device. In this case, when you enable query mode, the stored certificates and CRLs are deleted from the device after you save the configuration. (If you copy the configuration to a TFTP site prior to enabling query mode, you can save any stored certificates and CRLs at the TFTP site.)

Before disabling query mode, perform the **copy system:running-config nvram:startup-config** command to save all current certificates and CRLs to NVRAM. Otherwise they could be lost during a reboot.

To specify that certificates and CRLs should not be stored locally on your device, but should be retrieved when required, enable query mode by using the following command in global configuration mode:

Note Query mode may affect availability if the CA is down.

SUMMARY STEPS

1. **crypto ca certificate query**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto ca certificate query** **Example:** `Device(config)# crypto ca certificate query`	Enables query mode, which causes certificates and CRLs not to be stored locally.

Configuring the Device Host Name and IP Domain Name

You must configure the host name and IP domain name of a device if this has not already been done. This is required because the device assigns a fully qualified domain name (FQDN) to the keys and certificates used by IPsec, and the FQDN is based on the host name and IP domain name assigned to the device. For example, a certificate named "device20.example.com" is based on a device host name of "device20" and a device IP domain name of "example.com".

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *name*
4. **ip domain-name** *name*
5. **end**

DETAILED STEPS

	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	**hostname** *name* **Example:** `Device(config)# hostname device1`	Configures the host name of the device.
Step 4	**ip domain-name** *name* **Example:** `Device(config)# ip domain-name domain.com`	Configures the IP domain name of the device.
Step 5	**end** **Example:** `Device(config)# end`	Exits global configuration and returns to privileged EXEC mode.

Generating an RSA Key Pair

Rivest, Shamir, and Adelman (RSA) key pairs are used to sign and encrypt IKE key management messages and are required before obtaining a certificate for your device.

SUMMARY STEPS

1. **enable**

2. **configure terminal**
3. **crypto key generate rsa** [**usage-keys**]
4. **end**

DETAILED STEPS

	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 key generate rsa** [**usage-keys**] **Example:** `Device(config)# crypto key generate rsa usage-keys`	Generates an RSA key pair. • Use the **usage-keys** keyword to specify special-usage keys instead of general-purpose keys.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration and returns to privileged EXEC mode.

Declaring a Certification Authority

You should declare one certification authority (CA) to be used by the device.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ca trustpoint** *name*
4. **enrollment url** *url*
5. **enrollment command**
6. **exit**
7. **crypto pki trustpoint** *name*
8. **crl query ldap**://*url*:[*port*]
9. **enrollment** {**mode ra** | **retry count** *number* | **retry period** *minutes* | **url** *url*}
10. **enrollment** {**mode ra** | **retry count** *number* | **retry period** *minutes* | **url** *url*}
11. **revocation-check** *method1* [*method2 method3*]
12. **end**

DETAILED STEPS

	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 ca trustpoint** *name* **Example:** `Device(config)# crypto ca trustpoint ka`	Declares the certification authority (CA) that your device should use and enters the CA profile enroll configuration mode.
Step 4	**enrollment url** *url* **Example:** `Device(ca-profile-enroll)# enrollment url http://entrust:81`	Specifies the URL of the CA server to which enrollment requests are sent.
Step 5	**enrollment command** **Example:** `Device(ca-profile-enroll)# enrollment command`	Specifies the HTTP command that is sent to the CA for enrollment.
Step 6	**exit** **Example:** `Device(ca-profile-enroll)# exit`	Exit CA profile enroll configuration mode and returns to global configuration mode.
Step 7	**crypto pki trustpoint** *name* **Example:** `Device(config)# crypto pki trustpoint ka`	Declares the trustpoint that your device should use and enters Ca-trustpoint configuration mode.
Step 8	**crl query ldap**://*url*:[*port*] **Example:** `Device(ca-trustpoint)# crl query ldap://bar.cisco.com:3899`	Queries the certificate revocation list (CRL) to ensure that the certificate of the peer is not revoked.
Step 9	**enrollment** {**mode ra** | **retry count** *number* | **retry period** *minutes* | **url** *url*} **Example:** `Device(ca-trustpoint)# enrollment retry period 2`	Specifies the enrollment wait period between certificate request retries.
Step 10	**enrollment** {**mode ra** | **retry count** *number* | **retry period** *minutes* | **url** *url*} **Example:** `Device(ca-trustpoint)# enrollment retry count 8`	Specifies the number of times a device will resend a certificate request when it does not receive a response from the previous request.

	Command or Action	Purpose
Step 11	**revocation-check** *method1* [*method2 method3*] **Example:** `Device(ca-trustpoint)# revocation-check crl ocsp`	Checks the revocation status of a certificate.
Step 12	**end** **Example:** `Device(ca-trustpoint)# end`	Exit CA trustpoint configuration mode and returns to privileged EXEC mode.

Configuring a Root CA (Trusted Root)

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ca trustpoint** *name*
4. **revocation-check** *method1* [*method2 method3*]
5. **root tftp** *server-hostname filename*
6. **enrollment http-proxy** *hostname port-number*
7. **end**

DETAILED STEPS

	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 ca trustpoint** *name* **Example:** `Device(config)# crypto ca trustpoint ka`	Declares the trustpoint that your device should use and enters CA trustpoint configuration mode.
Step 4	**revocation-check** *method1* [*method2 method3*] **Example:** `Device(ca-trustpoint)# revocation-check ocsp`	Checks the revocation status of a certificate.
Step 5	**root tftp** *server-hostname filename* **Example:** `Device(ca-trustpoint)# root tftp server1 file1`	Obtains the certification authority (CA) certificate via TFTP.

	Command or Action	Purpose
Step 6	**enrollment http-proxy** *hostname port-number* **Example:** `Device(ca-trustpoint)# enrollment http-proxy host2 8080`	Accesses the certification authority (CA) by HTTP through the proxy server.
Step 7	**end** **Example:** `Device(ca-trustpoint)# end`	Exits CA trustpoint configuration mode and returns to privileged EXEC mode.

Authenticating the CA

The device must authenticate the certification authority (CA). It does this by obtaining the self-signed certificate of the CA, which contains the public key of the CA. Because the certificate of the CA is self-signed (the CA signs its own certificate) the public key of the CA should be manually authenticated by contacting the CA administrator to compare the fingerprint of the CA certificate when you perform this step.

Perform the following task to get the public key of the CA:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki authenticate***name*
4. **end**

DETAILED STEPS

	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 authenticate***name* **Example:** `Device(config)# crypto pki authenticate myca`	Authenticates the CA by getting the certificate of the CA.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Requesting Signed Certificates

You must obtain a signed certificate from the certification authority (CA) for each of the RSA key pairs on your device. If you generated general-purpose RSA keys, your device has only one RSA key pair and needs only one certificate. If you previously generated special-usage RSA keys, your device has two RSA key pairs and needs two certificates.

Perform the following task to request signed certificates from the CA:

Note If your device reboots after you have issued the **crypto pki enroll** command, but before you have received the certificates, you must reissue the command and notify the CA administrator.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki enroll** *number*
4. **end**

DETAILED STEPS

	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 enroll** *number* **Example:** `Device(config)# crypto pki enroll myca`	Obtains certificates for your device from the CA.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

What to do next

Saving Your Configuration

Always remember to save your work when you make configuration changes.

Use the **copy system:running-config nvram:startup-config** command to save your configuration. This command includes saving RSA keys to private NVRAM. RSA keys are not saved with your configuration when you use a **copy system:running-config rcp:** or copy **system:running-config tftp:** command.

Monitoring and Maintaining Certification Authority

Requesting a Certificate Revocation List

You can request a certificate revocation list (CRL) only if the certification authority (CA) does not support a registration authority (RA). The following task applies only when the CA does not support an RA.

When a device receives a certificate from a peer, your device will download a CRL from the CA. The device then checks the CRL to make sure the certificate that the peer sent has not been revoked. (If the certificate appears on the CRL, the device will not accept the certificate and will not authenticate the peer.)

A CRL can be reused with subsequent certificates until the CRL expires if query mode is off. If the device receives a peer's certificate after the applicable CRL has expired, the device will download the new CRL.

If the device has a CRL that has not yet expired, but you suspect that the contents of the CRL are out of date, you can request that the latest CRL be downloaded immediately to replace the old CRL.

-

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki crl request** *name*
4. **end**

DETAILED STEPS

	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 crl request** *name* **Example:** `Device(config)# crypto pki crl request myca`	Requests that a new certificate revocation list (CRL) be obtained immediately from the CA.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

Querying a Certification Revocation List

You can query a certificate revocation list (CRL) only when you configure your device with a trusted root. When your device receives a certificate from a peer from another domain (with a different CA), the CRL downloaded from the CA of the device will not include certificate information about the peer. Therefore, you should check the CRL published by the configured root with the LDAP URL to ensure that the certificate of the peer has not been revoked.

If you would like CRL of the root certificate to be queried when the device reboots, you must enter the **crl query** command.

Perform the following task to query the CRL published by the configured root with the LDAP URL:

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *name*
4. **crl query ldap** ://*url* : [*port*]
5. **end**

DETAILED STEPS

	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** *name* **Example:** `Device(ca-trustpoint)# crypto pki trustpoint mytp`	Declares the trustpoint that your device should use and enters CA trustpoint configuration mode.
Step 4	**crl query ldap** ://*url* : [*port*] **Example:** `Device(ca-trustpoint)# crl query ldap://url:[port]`	Queries the CRL to ensure that the certificate of the peer has not been revoked.
Step 5	**end** **Example:** `Device(ca-trustpoint)# end`	Exits CA trustpoint configuration mode and returns to privileged EXEC mode.

Deleting RSA Keys from a Device

Under certain circumstances you may want to delete RSA keys from your device. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.

]

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key zeroize rsa** [*key-pair-label*]
4. **end**

DETAILED STEPS

	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 key zeroize rsa** [*key-pair-label*] **Example:** `Device(config)# crypto key zeroize rsa`	Deletes all Rivest, Shamir, and Adelman (RSA) keys from your device.
Step 4	**end** **Example:** `Device(config)# end`	Exits global configuration mode and returns to privileged EXEC mode.

What to do next

After you delete RSA keys from the device, you should also complete the following two additional tasks:

- Ask the CA administrator to revoke the device certificates at the CA; you must supply the challenge password that you created when you originally obtained the device certificates with the **crypto pki enroll** command.
- Manually remove the device certificates from the device configuration.

Deleting Public Keys for a Peer

Under certain circumstances you may want to delete RSA public keys of peer devices from your device configuration. For example, if you no longer trust the integrity of the public key of a peer, you should delete the key.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key pubkey-chain rsa**

4. **no named key** *key-name* [**encryption** | **signature**]
5. **end**

DETAILED STEPS

	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 key pubkey-chain rsa** **Example:** `Device(config)# crypto key pubkey-chain rsa`	Enters public key chain configuration mode, so that you can manually specify other devices' RSA public keys.
Step 4	**no named key** *key-name* [**encryption**	**signature**] **Example:** `Device(config-pubkey-c)# no named-key otherpeer.example.com`	Deletes the RSA public key of a remote peer and enters public key configuration mode.
Step 5	**end** **Example:** `Device(config-pubkey)# end`	Exits public key configuration mode and returns to privileged EXEC mode.

Deleting Certificates from the Configuration

If the need arises, you can delete certificates that are saved in your device. Your devices saves its own certificates, the certificate of the CA, and any RA certificates .

To delete the CA's certificate, you must remove the entire CA identity, which also removes all certificates associated with the CA—your router's certificate, the CA certificate, and any RA certificates.

SUMMARY STEPS

1. **enable**
2. **show crypto pki certificates**
3. **configure terminal**
4. **crypto pki certificate chain** *name*
5. **no certificate** *certificate-serial-number*
6. **exit**
7. **no crypto pki import** *name* **certificate**
8. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show crypto pki certificates** **Example:** `Device# show crypto pki certificates`	Displays information about your device certificate, the certification authority (CA) certificate, and any registration authority (RA) certificates.
Step 3	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 4	**crypto pki certificate chain** *name* **Example:** `Device(config)# crypto pki certificate chain myca`	Enters certificate chain configuration mode.
Step 5	**no certificate** *certificate-serial-number* **Example:** `Device(config-cert-chain)# no certificate 0123456789ABCDEF0123456789ABCDEF`	Deletes the certificate.
Step 6	**exit** **Example:** `Device(config-cert-chain)# exit`	Exits certificate chain configuration mode and returns to global configuration mode.
Step 7	**no crypto pki import** *name* **certificate** **Example:** `Device(config)# no crypto pki import MS certificate`	Deletes a certificate manually.
Step 8	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Viewing Keys and Certificates

Perform the following task toview keys and certificates:

SUMMARY STEPS

1. **enable**
2. **show crypto key mypubkey rsa** [*keyname*]
3. **show crypto key pubkey-chain rsa**
4. **show crypto key pubkey-chain rsa** [**name** *key-name* | **address** *key-address*]

5. **show crypto pki certificates**
6. **show crypto pki trustpoints**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show crypto key mypubkey rsa** [*keyname*] **Example:** `Device# show crypto key mypubkey rsa [keyname]`	Displays the RSA public keys configured on a device.
Step 3	**show crypto key pubkey-chain rsa** **Example:** `Device# show crypto key pubkey-chain rsa`	Displays the RSA public keys of the peer that are stored on a device.
Step 4	**show crypto key pubkey-chain rsa** [**name** *key-name* | **address** *key-address*] **Example:** `Device# show crypto key pubkey-chain rsa address 209.165.202.129`	Displays the address of a specific key.
Step 5	**show crypto pki certificates** **Example:** `Device# show crypto pki certificates`	Displays information about the device certificate, the certification authority (CA) certificate, and any registration authority (RA) certificates
Step 6	**show crypto pki trustpoints** **Example:** `Device# show crypto pki certificates`	Displays trustpoints that are configured on a device.