Deploying RSA Keys Within a PKI

First Published: May 2, 2005
Last Updated: March 31, 2011

This module explains how to set up and deploy Rivest, Shamir, and Adelman (RSA) keys within a public key infrastructure (PKI). An RSA key pair (a public and a private key) is required before you can obtain a certificate for your router; that is, the end host must generate a pair of RSA keys and exchange the public key with the certification authority (CA) to obtain a certificate and enroll in a PKI.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for RSA Keys Within a PKI" section on page 24.

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

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Configuring RSA Keys for a PKI

- Before setting up and deploying RSA keys for a PKI, you should be familiar with the module *Cisco IOS PKI Overview: Understanding and Planning a PKI*.
- As of Cisco IOS Release 12.3(7)T, all commands that begin as "crypto ca" have been changed to begin as "crypto pki." Although the router will still accept crypto ca commands, all output will be read back as crypto pki.

Information About RSA Keys Configuration

RSA Keys Overview

An RSA key pair consists of a public key and a private key. When setting up your PKI, you must include the public key in the certificate enrollment request. After the certificate has been granted, the public key will be included in the certificate so that peers can use it to encrypt data that is sent to the router. The private key is kept on the router and used both to decrypt the data sent by peers and to digitally sign transactions when negotiating with peers.

RSA key pairs contain a key modulus value. The modulus determines the size of the RSA key. The larger the modulus, the more secure the RSA key. However, keys with large modulus values take longer to generate, and encryption and decryption operations take longer with larger keys.

Note As of Cisco IOS Release 12.4(11)T, peer *public* RSA key modulus values up to 4096 bits are automatically supported.

The largest private RSA key modulus is 4096 bits. Therefore, the largest RSA private key a router may generate or import is 4096 bits. However, RFC 2409 restricts the private key size to 2048 bits or less for RSA encryption.

The recommended modulus value for a CA is 2048 bits; the recommended modulus value for a client is 1024 bits.

Usage RSA Keys Versus General-Purpose RSA Keys

There are two mutually exclusive types of RSA key pairs—usage keys and general-purpose keys. When you generate RSA key pairs (via the **crypto key generate rsa** command), you will be prompted to select either usage keys or general-purpose keys.

Usage RSA Keys

Usage keys consist of two RSA key pairs—one RSA key pair is generated and used for encryption and one RSA key pair is generated and used for signatures. With usage keys, each key is not unnecessarily exposed. (Without usage keys, one key is used for both authentication methods, increasing the exposure of that key.)

General-Purpose RSA Keys

General-purpose keys consist of only one RSA key pair that used for both encryption and signatures. General-purpose key pairs are used more frequently than usage key pairs.

How RSA Key Pairs are Associated with a Trustpoint

A trustpoint, also known as the certificate authority (CA), manages certificate requests and issues certificates to participating network devices. These services provide centralized key management for the participating devices and are explicitly trusted by the receiver to validate identities and to create digital certificates. Before any PKI operations can begin, the CA generates its own public key pair and creates a self-signed CA certificate; thereafter, the CA can sign certificate requests and begin peer enrollment for the PKI.

Reasons to Store Multiple RSA Keys on a Router

Configuring multiple RSA key pairs allows the Cisco IOS software to maintain a different key pair for each CA with which it is dealing or the software can maintain multiple key pairs and certificates with the same CA. Thus, the Cisco IOS software can match policy requirements for each CA without compromising the requirements specified by the other CAs, such as key length, key lifetime, and general-purpose versus usage keys.

Named key pairs (which are specified via the **label** *key-label* option) allow you to have multiple RSA key pairs, enabling the Cisco IOS software to maintain a different key pair for each identity certificate.

Benefits of Exportable RSA Keys

Caution Exportable RSA keys should be carefully evaluated before use because using exportable RSA keys introduces the risk that these keys might be exposed.

Any existing RSA keys are *not* exportable. New keys are generated as nonexportable by default. It is not possible to convert an existing nonexportable key to an exportable key.

As of Cisco IOS Release 12.2(15)T, users can share the private RSA key pair of a router with standby routers, therefore transferring the security credentials between networking devices. The key pair that is shared between two routers will allow one router to immediately and transparently take over the functionality of the other router. If the main router were to fail, the standby router could be dropped into the network to replace the failed router without the need to regenerate keys, reenroll with the CA, or manually redistribute keys.

Exporting and importing an RSA key pair also enables users to place the same RSA key pair on multiple routers so that all management stations using Secure Shell (SSH) can be configured with a single public RSA key.

Exportable RSA Keys in PEM-Formatted Files

Using privacy-enhanced mail (PEM)-formatted files to import or export RSA keys can be helpful for customers who are running Cisco IOS software Release 12.3(4)T or later and who are using secure socket layer (SSL) or secure shell (SSH) applications to manually generate RSA key pairs and import the keys back into their PKI applications. PEM-formatted files allow customers to directly use existing RSA key pairs on their Cisco IOS routers instead of generating new keys.

Passphrase Protection While Importing and Exporting RSA Keys

You have to include a passphrase to encrypt the PKCS12 file or the PEM file that will be exported, and when the PKCS12 or PEM file is imported, the same passphrase has to be entered to decrypt it. Encrypting the PKCS12 or PEM file when it is being exported, deleted, or imported protects the file from unauthorized access and use while it is being transported or stored on an external device.

The passphrase can be any phrase that is at least eight characters in length; it can include spaces and punctuation, excluding the question mark (?), which has special meaning to the Cisco IOS parser.

How to Convert an Exportable RSA Key Pair to a Nonexportable RSA Key Pair

Passphrase protection protects the external PKCS12 or PEM file from unauthorized access and use. To prevent an RSA key pair from being exported, it must be labeled “nonexportable.” To convert an exportable RSA key pair into a nonexportable key pair, the key pair must be exported and then reimported without specifying the “exportable” keyword.

How to Set Up and Deploy RSA Keys Within a PKI

Generating an RSA Key Pair

Perform this task to manually generate an RSA key pair.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename:*] [**on** *devicename:*]
4. **exit**
5. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename:*] [**on** *devicename:*] **Example:** `Router(config)# crypto key generate rsa general-keys modulus 360`	(Optional) Generates the RSA key pair for the certificate server. • The **storage** keyword specifies the key storage location. • When specifying a label name by specifying the *key-label* argument, you must use the same name for the label that you plan to use for the certificate server (through the **crypto pki server** *cs-label* command). If a *key-label* argument is not specified, the default value, which is the fully qualified domain name (FQDN) of the router, is used. If the exportable RSA key pair is manually generated after the CA certificate has been generated, and before issuing the **no shutdown** command, then use the **crypto ca export pkcs12** command to export a PKCS12 file that contains the certificate server certificate and the private key. • By default, the modulus size of a CA key is 1024 bits. The recommended modulus for a CA key is 2048 bits. The range for a modulus size of a CA key is from 350 to 4096 bits. • The **on** keyword specifies that the RSA key pair is created on the specified device, including a Universal Serial Bus (USB) token, local disk, or NVRAM. The name of the device is followed by a colon (:). **Note** Keys created on a USB token must be 2048 bits or less.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys of your router. This step allows you to verify that the RSA key pair has been successfully generated.

What to Do Next

After you have successfully generated an RSA key pair, you can proceed to any of the additional tasks in this module to generate additional RSA key pairs, perform export and import of RSA key pairs, or configure additional security parameters for the RSA key pair (such as encrypting or locking the private key).

Managing RSA Key Pairs and Trustpoint Certificates

Perform this task to configure the router to generate and store multiple RSA key pairs, associate the key pairs with a trustpoint, and get the certificates for the router from the trustpoint.

Prerequisites

You must have already generated an RSA key pair as shown in the task "Generating an RSA Key Pair."

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki trustpoint** *name*
4. **rsakeypair** *key-label* [*key-size* [*encryption-key-size*]]
5. **enrollment selfsigned** (Optional)
6. **subject-alt-name** *name* (Optional)
7. **exit**
8. **cypto pki enroll** *name*
9. **exit**
10. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto pki trustpoint** *name* **Example:** `Router(config)# crypto pki trustpoint TESTCA`	Creates a trustpoint and enters ca-trustpoint configuration mode.
Step 4	**rsakeypair** *key-label* [*key-size* [*encryption-key-size*]] **Example:** `Router(ca-trustpoint)# rsakeypair fancy-keys`	(Optional) The *key-label* argument specifies the name of the RSA key pair generated during enrollment (if it does not already exist or if the **auto-enroll regenerate** command is configured) to be used with the trustpoint certificate. By default, the fully qualified domain name (FQDN) key is used. • (Optional) The *key-size* argument specifies the size of the RSA key pair. • (Optional) The *encryption-key-size* argument specifies the size of the second key, which is used to request separate encryption, signature keys, and certificates.
Step 5	**enrollment selfsigned** **Example:** `Router(ca-trustpoint)# enrollment selfsigned`	(Optional) Specifies self-signed enrollment for a trustpoint.
Step 6	**subject-alt-name** *name* **Example:** `Router(ca-trustpoint)# subject-alt-name TESTCA`	(Optional) The *name* argument specifies the trustpoint's name in the Subject Alternative Name (subjectAltName) field in the X.509 certificate, which is contained in the trustpoint certificate. By default, the Subject Alternative Name field is not included in the certificate. **Note** This X.509 certificate field is defined in RFC 2511. This option is used to create a self-signed trustpoint certificate for the router that contains the trustpoint name in the Subject Alternative Name (subjectAltName) field. This Subject Alternative Name can be used only when the **enrollment selfsigned** command is specified for self-signed enrollment in the trustpoint policy.
Step 7	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode.

	Command or Action	Purpose

Step 8	**cypto pki enroll** *name*
	
	Example:
	`Router(config)# cypto pki enroll TESTCA`
	`% Include the router serial number in the subject name? [yes/no]: no`
	`% Include an IP address in the subject name? [no]:`
	`Generate Self Signed Router Certificate? [yes/no]: yes`
	
	`Router Self Signed Certificate successfully created`
		Requests the certificates for the router from the trustpoint.
		The *name* argument specifies the trustpoint name. Once this command is entered, answer the prompts.
		Note Use the same trustpoint name entered with the **crypto pki trustpoint** command.

Step 9	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 10	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys of your router. This step allows you to verify that the RSA key pair has been successfully generated.

Example

The following example shows how to create a self-signed trustpoint certificate for the router that contains the trustpoint name in the Subject Alternative Name (subjectAltName) field:

```
Router> enable
Router# configure terminal
Router(config)# crypto pki trustpoint TESTCA
Router(ca-trustpoint)# enrollment selfsigned
Router(ca-trustpoint)# subject-alt-name TESTCA
Router(ca-trustpoint)# exit
Router(config)# cypto pki enroll TESTCA
% Include the router serial number in the subject name? [yes/no]: no
% Include an IP address in the subject name? [no]:
Generate Self Signed Router Certificate? [yes/no]: yes

Router Self Signed Certificate successfully created
Router(config)# exit
```

The following certificate is created:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: CN=TESTCA/unstructuredName=r1.cisco.com
        Validity
            Not Before: Mar 22 20:26:20 2010 GMT
            Not After : Jan  1 00:00:00 2020 GMT
        Subject: CN=TESTCA/unstructuredName=r1.cisco.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (512 bit)
```

```
                Modulus (512 bit):
                    00:8d:71:2e:3b:eb:a2:e2:f3:44:d9:bc:a9:85:88:
                    f4:a9:bd:c9:7f:f0:69:f5:e7:75:8f:00:f2:8e:3e:
                    2f:ca:5e:c5:08:43:95:8c:a2:6a:ae:ce:a0:ae:82:
                    61:61:ff:4e:8c:8f:89:d1:56:d8:35:34:b7:95:93:
                    1a:72:03:71:fb
                Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Basic Constraints: critical
                CA:TRUE
                X509v3 Subject Alternative Name:
                DNS:TESTCA
                X509v3 Authority Key Identifier:
                keyid:F9:A4:95:87:5F:A4:CA:7D:65:FA:BE:38:20:55:18:F9:4C:6C:D5:F3

                X509v3 Subject Key Identifier:
                F9:A4:95:87:5F:A4:CA:7D:65:FA:BE:38:20:55:18:F9:4C:6C:D5:F3
        Signature Algorithm: md5WithRSAEncryption
            6d:92:e7:a8:a5:1a:5a:ef:13:58:02:1b:79:17:93:41:37:c9:
            2d:9f:1a:a3:f5:3a:73:05:cd:d1:02:84:43:7e:e0:84:07:46:
            55:f9:45:59:51:ba:25:48:6f:d8:e1:0d:35:44:07:5c:16:17:
            35:45:99:e2:80:6e:53:e5:35:76
-----BEGIN CERTIFICATE-----
MIIBszCCAV2gAwIBAgIBAjANBgkqhkiG9w0BAQQFADAuMQ8wDQYDVQQDEwZURVNU
Q0ExGzAZBgkqhkiG9w0BCQIWDHIxLmNpc2NvLmNvbTAeFw0xMDAzMjIyMDI2MjBa
Fw0yMDAxMDEwMDAwMDBaMC4xDzANBgNVBAMTBlRFU1RDQTEbMBkGCSqGSIb3DQEJ
AhYMcjEuY2lzY28uY29tMFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAI1xLjvrouLz
RNm8qYWI9Km9yX/wafXndY8A8o4+L8pexQhDlYyiaq7OoK6CYWH/ToyPidFW2DU0
t5WTGnIDcfsCAwEAAaNmMGQwDwYDVR0TAQH/BAUwAwEB/zARBgNVHREECjAIggZU
RVNUQ0EwHwYDVR0jBBgwFoAU+aSVh1+kyn1l+r44IFUY+Uxs1fMwHQYDVR0OBBYE
FPmklYdfpMp9Zfq+OCBVGPlMbNXzMA0GCSqGSIb3DQEBBAUAA0EAbZLnqKUaWu8T
WAIbeReTQTfJLZ8ao/U6cwXN0QKEQ37ghAdGVflFWVG6JUhv2OENNUQHXBYXNUWZ
4oBuU+U1dg==
-----END CERTIFICATE-----
```

Exporting and Importing RSA Keys

This section contains the following tasks that can be used for exporting and importing RSA keys. Whether you are using PKCS12 files or PEM files, exportable RSA keys allow you to use existing RSA keys on Cisco IOS routers instead of having to generate new RSA keys if the main router were to fail.

- Exporting and Importing RSA Keys in PKCS12 Files, page 9
- Exporting and Importing RSA Keys in PEM-Formatted Files, page 12

Exporting and Importing RSA Keys in PKCS12 Files

Exporting and importing RSA key pairs enables users to transfer security credentials between devices. The key pair that is shared between two devices allows one device to immediately and transparently take over the functionality of the other router.

Prerequisites for Exporting and Importing RSA Key in PKCS12 Files

You must generate an RSA key pair and mark it "exportable" as specified in the task "Generating an RSA Key Pair."

Restrictions for Exporting and Importing RSA Keys in PKCS12 Files

- You cannot export RSA keys that existed on the router before your system was upgraded to Cisco IOS Release 12.2(15)T or later. You have to generate new RSA keys and label them as "exportable" after you upgrade the Cisco IOS software.
- When you import a PKCS12 file that was generated by a third-party application, the PKCS12 file must include a CA certificate.
- If you want reexport an RSA key pair after you have already exported the key pair and imported them to a target router, you must specify the **exportable** keyword when you are importing the RSA key pair.
- The largest RSA key a router may import is 2048-bits.

SUMMARY STEPS

1. **crypto pki trustpoint** *name*
2. **rsakeypair** *key-label* [*key-size* [*encryption-key-size*]]
3. **exit**
4. **crypto pki export** *trustpointname* **pkcs12** *destination-url passphrase*
5. **crypto pki import** *trustpointname* **pkcs12** *source-url passphrase*
6. **exit**
7. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto pki trustpoint** *name* **Example:** `Router(config)# crypto pki trustpoint my-ca`	Creates the trustpoint name that is to be associated with the RSA key pair and enters ca-trustpoint configuration mode.
Step 2	**rsakeypair** *key-label* [*key-size* [*encryption-key-size*]] **Example:** `Router(ca-trustpoint)# rsakeypair my-keys`	Specifies the key pair that is to be used with the trustpoint.
Step 3	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode.
Step 4	**crypto pki export** *trustpointname* **pkcs12** *destination-url passphrase* **Example:** `Router(config)# crypto pki export my-ca pkcs12 tftp://tftpserver/my-keys PASSWORD`	Exports the RSA keys via the trustpoint name. **Note** You can export the trustpoint using any of the following file system types: flash, FTP, null, NVRAM, remote file copying (RCP), SCP, system, TFTP, Webflash, Xmodem, or Ymodem.

	Command or Action	Purpose
Step 5	**crypto pki import** *trustpointname* **pkcs12** *source-url passphrase* **Example:** `Router(config)# crypto pki import my-ca pkcs12 tftp://tftpserver/my-keys PASSWORD`	Imports the RSA keys to the target router.
Step 6	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 7	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys of your router.

Exporting and Importing RSA Keys in PEM-Formatted Files

Perform this task to export or import RSA key pairs in PEM files.

Prerequisites for Exporting and Importing RSA Keys in PEM-Formatted Files

You must generate an RSA key pair and mark it "exportable" as specified in the task "Generating an RSA Key Pair."

Restrictions for Exporting and Importing RSA Keys in PEM Formatted Files

- You cannot export and import RSA keys that were generated without an exportable flag before your system was upgraded to Cisco IOS Release 12.3(4)T or a later release. You have to generate new RSA keys after you upgrade the Cisco IOS software.
- The largest RSA key a router may import is 2048 bits.

SUMMARY STEPS

1. **crypto key generate rsa** {**usage-keys** | **general-keys**} **label** *key-label* [**exportable**]
2. **crypto key export rsa** *key-label* **pem** {**terminal** | **url** *url*} {**3des** | **des**} *passphrase*
3. **crypto key import rsa** *key-label* **pem** [**usage-keys**] {**terminal** | **url** *url*} [**exportable**] *passphrase*
4. **exit**
5. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto key generate rsa** {**usage-keys** | **general-keys**} **label** *key-label* [**exportable**] **Example:** `Router(config)# crypto key generate rsa general-keys label mykey exportable`	Generates RSA key pairs. To use PEM files, the RSA key pair must be labeled exportable.
Step 2	**crypto key export rsa** *key-label* **pem** {**terminal** | **url** *url*} {**3des** | **des**} *passphrase* **Example:** `Router(config)# crypto key export rsa mycs pem url nvram: 3des PASSWORD`	Exports the generated RSA key pair. **Tip** Be sure to keep the PEM file safe. For example, you may want to store it on another backup router.
Step 3	**crypto key import rsa** *key-label* **pem** [**usage-keys**] {**terminal** | **url** *url*} [**exportable**] *passphrase* **Example:** `Router(config)# crypto key import rsa mycs2 pem url nvram: PASSWORD`	Imports the generated RSA key pair. **Note** If you do not want the key to be exportable from your CA, import it back to the CA after it has been exported as a nonexportable key pair. Thus, the key cannot be taken off again.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys of your router.

Encrypting and Locking Private Keys on a Router

Digital signatures are used to authenticate one device to another device. To use digital signatures, private information (the private key) must be stored on the device that is providing the signature. The stored private information may aid an attacker who steals the hardware device that contains the private key; for example, a thief might be able to use the stolen router to initiate a secure connection to another site by using the RSA private keys stored in the router.

Note RSA keys are lost during password recovery operations. If you lose your password, the RSA keys will be deleted when you perform the password recovery operation. (This function prevents an attacker from performing password recovery and then using the keys.)

To protect the private RSA key from an attacker, a user can encrypt the private key that is stored in NVRAM via a passphrase. Users can also “lock” the private key, which blocks new connection attempts from a running router and protects the key in the router if the router is stolen by an attempted attacker.

Perform this task to encrypt and lock the private key that is saved to NVRAM.

Note The RSA keys must be unlocked while enrolling the CA. The keys can be locked while authenticating the router with the CA because the private key of the router is not used during authentication.

Prerequisites

Before encrypting or locking a private key, you should perform the following tasks:

- Generate an RSA key pair as shown in the task "Generating an RSA Key Pair."
- Optionally, you can authenticate and enroll each router with the CA server.

Restrictions for Encrypting and Locking Private Keys

Backward Compatibility Restriction

Any image prior to Cisco IOS Release 12.3(7)T does not support encrypted keys. To prevent your router from losing all encrypted keys, ensure that only unencrypted keys are written to NVRAM before booting an image prior to Cisco IOS Release 12.3(7)T.

If you must download an image prior to Cisco IOS Release 12.3(7)T, decrypt the key and immediately save the configuration so the downloaded image does not overwrite the configuration.

Interaction with Applications

An encrypted key is not effective after the router boots up until you manually unlock the key (via the **crypto key unlock rsa** command). Depending on which key pairs are encrypted, this functionality may adversely affect applications such as IP security (IPsec), SSH, and SSL; that is, management of the router over a secure channel may not be possible until the necessary key pair is unlocked.

SUMMARY STEPS

1. **crypto key encrypt** [**write**] **rsa** [**name** *key-name*] **passphrase** *passphrase*
2. **exit**
3. **show crypto key mypubkey rsa**
4. **crypto key lock rsa** [**name** *key-name*] **passphrase** *passphrase*
5. **show crypto key mypubkey rsa**
6. **crypto key unlock rsa** [**name** *key-name*] **passphrase** *passphrase*
7. **configure terminal**
8. **crypto key decrypt** [**write**] **rsa** [**name** *key-name*] **passphrase** *passphrase*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**crypto key encrypt** [**write**] **rsa** [**name** *key-name*] **passphrase** *passphrase* **Example:** `Router(config)# crypto key encrypt write rsa name pki.example.com passphrase password`	Encrypts the RSA keys. After this command is issued, the router can continue to use the key; the key remains unlocked. **Note** If the **write** keyword is not issued, the configuration must be manually written to NVRAM; otherwise, the encrypted key will be lost next time the router is reloaded.
Step 2	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 3	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Shows that the private key is encrypted (protected) and unlocked. **Note** You can also use this command to verify that applications such as Internet Key Exchange (IKE) and SSH are properly working after the key has been encrypted.
Step 4	**crypto key lock rsa** [**name** *key-name*] **passphrase** *passphrase* **Example:** `Router# crypto key lock rsa name pki.example.com passphrase password`	(Optional) Locks the encrypted private key on a running router. **Note** After the key is locked, it cannot be used to authenticate the router to a peer device. This behavior disables any IPSec or SSL connections that use the locked key. Any existing IPSec tunnels created on the basis of the locked key will be closed. If all RSA keys are locked, SSH will automatically be disabled.
Step 5	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Shows that the private key is protected and locked. The output will also show failed connection attempts via applications such as IKE, SSH, and SSL.
Step 6	**crypto key unlock rsa** [**name** *key-name*] **passphrase** *passphrase* **Example:** `Router# crypto key unlock rsa name pki.example.com passphrase password`	(Optional) Unlocks the private key. **Note** After this command is issued, you can continue to establish IKE tunnels.

	Command or Action	Purpose
Step 7	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 8	**crypto key decrypt** [**write**] **rsa** [**name** *key-name*] **passphrase** *passphrase* **Example:** `Router(config)# crypto key decrypt write rsa name pki.example.com passphrase password`	(Optional) Deletes the encrypted key and leaves only the unencrypted key. **Note** The **write** keyword immediately saves the unencrypted key to NVRAM. If the **write** keyword is not issued, the configuration must be manually written to NVRAM; otherwise, the key will remain encrypted the next time the router is reloaded.

Removing RSA Key Pair Settings

An RSA key pair may need to be removed for one of the following reasons:

- During manual PKI operations and maintenance, old RSA keys can be removed and replaced with new keys.
- An existing CA is replaced and the new CA requires newly generated keys; for example, the required key size might have changed in an organization so you would have to delete the old 1024-bit keys and generate new 2048-bit keys.
- The peer router's public keys can be deleted in order to help debug signature verification problems in IKEv1 and IKEv2. Keys are cached by default with the lifetime of the certificate revocation list (CRL) associated with the trustpoint.

Perform this task to remove all RSA keys or the specified RSA key pair that has been generated by your router.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto key zeroize rsa** [*key-pair-label*]
4. **crypto key zeroize pubkey-chain** [*index*]
5. **exit**
6. **show crypto key mypubkey rsa**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto key zeroize rsa** [*key-pair-label*] **Example:** `Router(config)# crypto key zeroize rsa fancy-keys`	Deletes RSA key pairs from your router. • If the *key-pair-label* argument is not specified, all RSA keys that have been generated by your router will be deleted.
Step 4	**crypto key zeroize pubkey-chain** [*index*] **Example:** `Router(config)# crypto key zeroize pubkey-chain`	Deletes the remote peer's public key from the cache. (Optional) Use the *index* argument to delete a particular public key index entry. If no index entry is specified, then all the entries are deleted. The acceptable range of index entries is from 1 to 65535.
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**show crypto key mypubkey rsa** **Example:** `Router# show crypto key mypubkey rsa`	(Optional) Displays the RSA public keys of your router. This step allows you to verify that the RSA key pair has been successfully generated.

Configuration Examples for RSA Key Pair Deployment

Generating and Specifying RSA Keys: Example

The following example is a sample trustpoint configuration that shows how to generate and specify the RSA key pair "exampleCAkeys":

```
crypto key generate rsa general-purpose exampleCAkeys
crypto ca trustpoint exampleCAkeys
 enroll url http://exampleCAkeys/certsrv/mscep/mscep.dll
 rsakeypair exampleCAkeys 1024 1024
```

Exporting and Importing RSA Keys: Examples

Exporting and Importing RSA Keys in PKCS12 Files: Example

In the following example, an RSA key pair “mynewkp” is generated on Router A, and a trustpoint name “mynewtp” is created and associated with the RSA key pair. The trustpoint is exported to a TFTP server, so that it can be imported on Router B. By importing the trustpoint “mynewtp” to Router B, the user has imported the RSA key pair “mynewkp” to Router B.

Router A

```
crypto key generate rsa general label mykeys exportable
! The name for the keys will be:mynewkp
Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take
a few minutes.
How many bits in the modulus [512]:
% Generating 512 bit RSA keys ...[OK]
!
crypto pki trustpoint mynewtp
 rsakeypair mykeys
 exit

crypto pki export mytp pkcs12 flash:myexport companyname
Destination filename [myexport]?
Writing pkcs12 file to tftp:/mytftpserver/myexport
CRYPTO_PKI:Exported PKCS12 file successfully.
Verifying checksum... OK (0x3307)
!
Feb 18 17:30:09 GMT:%CRYPTO-6-PKCS12EXPORT_SUCCESS:PKCS #12 Successfully Exported.
```

Router B

```
crypto pki import mynewtp pkcs12 flash:myexport companyname
Source filename [myexport]?
CRYPTO_PKI:Imported PKCS12 file successfully.

!
Feb 18 18:07:50 GMT:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
```

Generating, Exporting, Importing, and Verifying RSA Keys in PEM Files: Example

The following example shows how to generate, export, bring the key back (import), and verify the status of the RSA key pair “mycs”:

```
! Generate the key pair
!
Router(config)# crypto key generate rsa general-purpose label mycs exportable
The name for the keys will be: mycs

Choose the size of the key modulus in the range of 360 to 2048 for your
General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.
```

```
How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys ...[OK]
!
! Archive the key pair to a remote location, and use a good password.
!
Router(config)# crypto key export rsa mycs pem url nvram:3des PASSWORD

% Key name:mycs
Usage:General Purpose Key
Exporting public key...
Destination filename [mycs.pub]?
Writing file to nvram:mycs.pub
Exporting private key...
Destination filename [mycs.prv]?
Writing file to nvram:mycs.prv
!
! Import the key as a different name.
!
Router(config)# crypto key import rsa mycs2 pem url nvram:mycs PASSWORD

% Importing public key or certificate PEM file...
Source filename [mycs.pub]?
Reading file from nvram:mycs.pub
% Importing private key PEM file...
Source filename [mycs.prv]?
Reading file from nvram:mycs.prv% Key pair import succeeded.
!
! After the key has been imported, it is no longer exportable.
!
! Verify the status of the key.
!
Router# show crypto key mypubkey rsa

% Key pair was generated at:18:04:56 GMT Jun 6 2003
Key name:mycs
Usage:General Purpose Key
Key is exportable.
Key Data:
30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00E65253
9C30C12E 295AB73F B1DF9FAD 86F88192 7D4FA4D2 8BA7FB49 9045BAB9 373A31CB
A6B1B8F4 329F2E7E 8A50997E AADBCFAA 23C29E19 C45F4F05 DBB2FA51 4B7E9F79
A1095115 759D6BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7D8DC6A3 D31B2486
C9C96D2C 2F70B50D 3B4CDDAE F661041A 445AE11D 002EEF08 F2A627A0 5B020301 0001
% Key pair was generated at:18:17:25 GMT Jun 6 2003
Key name:mycs2
Usage:General Purpose Key
Key is not exportable.
Key Data:
30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00E65253
9C30C12E 295AB73F B1DF9FAD 86F88192 7D4FA4D2 8BA7FB49 9045BAB9 373A31CB
A6B1B8F4 329F2E7E 8A50997E AADBCFAA 23C29E19 C45F4F05 DBB2FA51 4B7E9F79
A1095115 759D6BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7D8DC6A3 D31B2486
C9C96D2C 2F70B50D 3B4CDDAE F661041A 445AE11D 002EEF08 F2A627A0 5B020301 0001
```

Exporting Router RSA Key Pairs and Certificates from PEM Files: Example

The following example shows how to generate and export the RSA key pair “aaa” and certificates of the router in PEM files that are associated with the trustpoint “mycs.” This example also shows PEM-formatted files, which include PEM boundaries before and after the base64-encoded data, that are used by other SSL and SSH applications.

```
Router(config)# crypto key generate rsa general-keys label aaa exportable
```

```
The name for the keys will be:aaa
Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose
Keys. Choosing a key modulus greater than 512 may take a few minutes.
!
How many bits in the modulus [512]:
% Generating 512 bit RSA keys ...[OK]
!
Router(config)# crypto pki trustpoint mycs
Router(ca-trustpoint)# enrollment url http://mycs
Router(ca-trustpoint)# rsakeypair aaa
Router(ca-trustpoint)# exit
Router(config)# crypto pki authenticate mycs
Certificate has the following attributes:
Fingerprint:C21514AC 12815946 09F635ED FBB6CF31
% Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.
!
Router(config)# crypto pki enroll mycs
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this password to the CA
Administrator in order to revoke your certificate.
For security reasons your password will not be saved in the configuration.
Please make a note of it.

Password:
Re-enter password:

% The fully-qualified domain name in the certificate will be: Router
% The subject name in the certificate will be:host.example.com
% Include the router serial number in the subject name? [yes/no]: n
% Include an IP address in the subject name? [no]: n
Request certificate from CA? [yes/no]: y
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.

Router(config)# Fingerprint:8DA777BC 08477073 A5BE2403 812DD157

00:29:11:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority

Router(config)# crypto ca export aaa pem terminal 3des password
% CA certificate:
-----BEGIN CERTIFICATE-----
MIICAzCCAa2gAwIBAgIBATANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJVUzES
<snip>
waDeNOSI3WlDa0AWq5DkVBkxwgn0TqIJXJOCttjHnWHK1LMcMVGn
-----END CERTIFICATE-----

% Key name:aaa
Usage:General Purpose Key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type:4,ENCRYPTED
DEK-Info:DES-EDE3-CBC,ED6B210B626BC81A

Urguv0jnjwOgowWVUQ2XR5nbzzYHI2vGLunpH/IxIsJuNjRVjbAAUpGk7VnPCT87
<snip>
kLCOtxzEv7JHc72gMku9uUlrLSnFH5slzAtoC0czfU4=
-----END RSA PRIVATE KEY-----

% Certificate:
-----BEGIN CERTIFICATE-----
MIICTjCCAfigAwIBAgICIQUwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
```

```
<snip>
6xlBaIsuMxnHmr89KkKkYlU6
-----END CERTIFICATE-----
```

Importing Router RSA Key Pairs and Certificate from PEM Files: Example

The following example shows how to import the RSA key pairs and certificate to the trustpoint "ggg" from PEM files via TFTP:

```
Router(config)# crypto pki import ggg pem url tftp://10.1.1.2/username/msca password
% Importing CA certificate...
Address or name of remote host [10.1.1.2]?
Destination filename [username/msca.ca]?
Reading file from tftp://10.1.1.2/username/msca.ca
Loading username/msca.ca from 10.1.1.2 (via Ethernet0):!
[OK - 1082 bytes]

% Importing private key PEM file...
Address or name of remote host [10.1.1.2]?
Destination filename [username/msca.prv]?
Reading file from tftp://10.1.1.2/username/msca.prv
Loading username/msca.prv from 10.1.1.2 (via Ethernet0):!
[OK - 573 bytes]

% Importing certificate PEM file...
Address or name of remote host [10.1.1.2]?
Destination filename [username/msca.crt]?
Reading file from tftp://10.1.1.2/username/msca.crt
Loading username/msca.crt from 10.1.1.2 (via Ethernet0):!
[OK - 1289 bytes]
% PEM files import succeeded.
Router(config)#
```

Encrypting and Locking Private Keys on a Router: Examples

Configuring and Verifying an Encrypted Key: Example

The following example shows how to encrypt the RSA key "pki-123.example.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the RSA key is encrypted (protected) and unlocked.

```
Router(config)# crypto key encrypt rsa name pki-123.example.com passphrase password
Router(config)# exit
Router# show crypto key mypubkey rsa

% Key pair was generated at:00:15:32 GMT Jun 25 2003
Key name:pki-123.example.com
Usage:General Purpose Key
*** The key is protected and UNLOCKED. ***
Key is not exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00E0CC9A 1D23B52C
CD00910C ABD392AE BA6D0E3F FC47A0EF 8AFEE340 0EC1E62B D40E7DCC
23C4D09E
03018B98 E0C07B42 3CFD1A32 2A3A13C0 1FF919C5 8DE9565F 1F020301 0001
```

```
% Key pair was generated at:00:15:33 GMT Jun 25 2003
Key name:pki-123.example.com.server
Usage:Encryption Key
Key is exportable.
Key Data:
307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00D3491E 2A21D383
854D7DA8 58AFBDAC 4E11A7DD E6C40AC6 66473A9F 0C845120 7C0C6EC8 1FFF5757
3A41CE04 FDCB40A4 B9C68B4F BC7D624B 470339A3 DE739D3E F7DDB549 91CD4DA4
DF190D26 7033958C 8A61787B D40D28B8 29BCD0ED 4E6275C0 6D020301 0001
Router#
```

Configuring and Verifying a Locked Key: Example

The following example shows how to lock the key "pki-123.example.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the key is protected (encrypted) and locked.

```
Router# crypto key lock rsa name pki-123.example.com passphrase password
!
Router# show crypto key mypubkey rsa

% Key pair was generated at:20:29:41 GMT Jun 20 2003
Key name:pki-123.example.com
Usage:General Purpose Key
*** The key is protected and LOCKED. ***
Key is exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00D7808D C5FF14AC
0D2B55AC 5D199F2F 7CB4B355 C555E07B 6D0DECBE 4519B1F0 75B12D6F 902D6E9F
B6FDAD8D 654EF851 5701D5D7 EDA047ED 9A2A619D 5639DF18 EB020301 0001
```

Where to Go Next

After you have generated an RSA key pair, you should set up the trustpoint. If you have already set up the trustpoint, you should authenticate and enroll the routers in a PKI. For information on enrollment, see the module "Configuring Certificate Enrollment for a PKI."

Additional References

Related Documents

Related Topic	Document Title
Overview of PKI, including RSA keys, certificate enrollment, and CAs	*Cisco IOS PKI Overview: Understanding and Planning a PKI*
PKI commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*

MIBs

MIBs	MIBs Link
None	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 2409	*The Internet Key Exchange (IKE)*
RFC 2511	Internet X.509 Certificate Request Message Format

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for RSA Keys Within a PKI

Table 1 lists the release history for this feature.

For information on a feature in this technology that is not documented here, see the "Implementing and Managing PKI Features Roadmap."

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 1 lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 1 Feature Information for RSA Keys Within a PKI

Feature Name	Software Releases	Feature Configuration Information
Cisco IOS 4096-Bit Public Key Support	12.4(12)T	This feature introduces Cisco IOS 4096-bit peer public key support. The following section provides information about this feature: • RSA Keys Overview
Exporting and Importing RSA Keys	12.2(15)T	This feature allows you to transfer security credentials between devices by exporting and importing RSA keys. The key pair that is shared between two devices will allow one device to immediately and transparently take over the functionality of the other router. The following sections provide information about this feature: • Benefits of Exportable RSA Keys • Exporting and Importing RSA Keys in PKCS12 Files The following commands were introduced or modified by this feature: **crypto ca export pkcs12**, **crypto ca import pkcs12**, **crypto key generate rsa (IKE)**

Table 1 Feature Information for RSA Keys Within a PKI (continued)

Feature Name	Software Releases	Feature Configuration Information
Import of RSA Key Pair and Certificates in PEM Format	12.3(4)T	This feature allows customers to use PEM-formatted files to import or export RSA key pairs. PEM-formatted files allow customers to directly use existing RSA key pairs on their Cisco IOS routers instead of generating new keys. The following sections provide information about this feature: • Benefits of Exportable RSA Keys • Exporting and Importing RSA Keys in PEM-Formatted Files The following commands were introduced by this feature: **crypto ca export pem**, **crypto ca import pem**, **crypto key export pem**, **crypto key import pem**
Multiple RSA Key Pair Support	12.2(8)T	This feature allows a user to configure a router to have multiple RSA key pairs. Thus, the Cisco IOS software can maintain a different key pair for each identity certificate. The following sections provide information about this feature: • Reasons to Store Multiple RSA Keys on a Router • Managing RSA Key Pairs and Trustpoint Certificates The following commands were introduced or modified by this feature: **crypto key generate rsa**, **crypto key zeroize rsa**, **rsakeypair**
Protected Private Key Storage	12.3(7)T	This feature allows a user to encrypt and lock the RSA private keys that are used on a Cisco IOS router, thereby, preventing unauthorized use of the private keys. The following section provides information about this feature: • Encrypting and Locking Private Keys on a Router The following commands were introduced or modified by this feature: **crypto key decrypt rsa**, **crypto key encrypt rsa**, **crypto key lock rsa**, **crypto key unlock rsa**, **show crypto key mypubkey rsa**

Table 1 Feature Information for RSA Keys Within a PKI (continued)

Feature Name	Software Releases	Feature Configuration Information
RSA 4096-bit Key Generation in Software Crypto Engine Support	15.1(1)T	The range value for the **modulus** keyword value for the **crypto key generate rsa** command is extended from 360 to 2048 bits to 360 to 4096 bits.
IOS PKI Performance Monitoring and Optimization	15.1(3)T	The IOS Performance Monitoring and Optimization feature provides a way to characterize the performance within the Public Key Infrastructure (PKI) subsystem and debug and analyze PKI performance related issues. This feature is discussed in further detail in the IOS Performance Monitoring and Optimization feature document. This feature also includes the following enhancements that can be found in this document: • A self-signed trustpoint certificate can be created for the router that contains the trustpoint name in the Subject Alternative Name (subjectAltName) field. • A peer router's public keys can be deleted to help debug signature verification problems in IKE version 1 and IKE version 2 and optimize the peer router's performance as a result of taking this action. These features can be found in the following sections: • "Generating an RSA Key Pair" section on page 4 • "Removing RSA Key Pair Settings" section on page 16 The following commands were introduced or modified by this feature: **crypto key zeroize pubkey-chain**, **subject-alt-name**