Deploying RSA Keys Within a PKI

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.

Security threats, as well as the cryptographic technologies to help protect against them, are constantly changing. For more information about the latest Cisco cryptographic recommendations, see the Next Generation Encryption (NGE) white paper.

Note

Finding Feature Information, page 1
Prerequisites for Configuring RSA Keys for a PKI, page 2
Information About RSA Keys Configuration, page 2
How to Set Up and Deploy RSA Keys Within a PKI, page 4
Configuration Examples for RSA Key Pair Deployment, page 18
Where to Go Next, page 22
Additional References, page 22
Feature Information for RSA Keys Within a PKI, page 23

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 table.

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

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 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.

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.
Do not manually generate an rsa keypair under trustpoint. If we want to manually generate the keys, generate the key pairs as usage-keys and not as general-purpose keys.

Caution

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. As a result, 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.

Exportable RSA keys 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**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Router&gt; enable</td>
<td>• Enter your password if prompted.</td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> crypto key generate rsa [general-keys</td>
<td>usage-keys</td>
</tr>
<tr>
<td>Example:</td>
<td>• The storage keyword specifies the key storage location.</td>
</tr>
<tr>
<td></td>
<td>• 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</td>
</tr>
</tbody>
</table>
### Purpose

- Command or Action
- Purpose

| Example: |
| Router(config)# crypto key generate rsa usage-keys modulus 2048 |

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 360 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.

**Caution**

- Do not manually generate an rsa keypair under trustpoint. If we want to manually generate the keys, generate the key pairs as usage-keys and not as general-purpose keys.

**Step 4**

| exit |
| Exits global configuration mode. |

| Example: |
| Router(config)# exit |

**Step 5**

| show crypto key mypubkey rsa |
| (Optional) Displays the RSA public keys of your router. |

| Example: |
| Router# show crypto key mypubkey rsa |

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.
Before You Begin
You must have already generated an RSA key pair as shown in the task “Generating an RSA Key Pair task.”

SUMMARY STEPS

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

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> enable</td>
<td>Enables privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Router&gt; enable</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> configure terminal</td>
<td>Enters global configuration mode.</td>
</tr>
<tr>
<td>Example: Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> crypto pki trustpoint name</td>
<td>Creates a trustpoint and enters ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td>Example: Router(config)# crypto pki trustpoint TESTCA</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> rsakeypair key-label [key-size [encryption-key-size]]</td>
<td>(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.</td>
</tr>
<tr>
<td>Example: Router(ca-trustpoint)# rsakeypair fancy-keys</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Optional) The key-size argument specifies the size of the RSA key pair. The recommended key size is 2048 bits.</td>
</tr>
</tbody>
</table>
### Command or Action | Purpose |
--- | --- |
| (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.

### Step 8 crypto pki enroll *name*

**Example:**
```
Router(config)# crypto pki enroll TESTCA
```

**Example:**
```
% Include the router serial number in the subject name? [yes/no]: no
```

**Example:**
```
% Include an IP address in the subject name? [no]:
```

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.
### Command or Action

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example: Generate Self Signed Router Certificate? [yes/no]: yes</td>
<td></td>
</tr>
<tr>
<td>Example: Router Self Signed Certificate successfully created</td>
<td></td>
</tr>
<tr>
<td><strong>Step 9</strong> exit</td>
<td>Exits global configuration mode.</td>
</tr>
<tr>
<td>Example: Router(config)# exit</td>
<td></td>
</tr>
<tr>
<td><strong>Step 10</strong> show crypto key mypubkey rsa</td>
<td>(Optional) Displays the RSA public keys of your router. This step allows you to verify that the RSA key pair has been successfully generated.</td>
</tr>
<tr>
<td>Example: Router# show crypto key mypubkey rsa</td>
<td></td>
</tr>
</tbody>
</table>

### 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)#hash sha256
Router(ca-trustpoint)#rsakeypair testca-rsa-key 2048
Router(ca-trustpoint)#exit
Router(config)#crypto pki enroll TESTCA
% Include the router serial number in the subject name? [yes/no]:no
% Include an IP address in the subject name? [no]: no
Generate Self Signed Router Certificate? [yes/no]: yes

Router Self Signed Certificate successfully created

Router(config)#
Router(config)#exit
Router#

The following certificate is created:

Router#show crypto pki certificate verbose
Router Self-Signed Certificate
Status: Available
Version: 3
Certificate Serial Number (hex): 01
Certificate Usage: General Purpose
Issuer:
  hostname=Router.cisco.com
Subject:
  Name: Router.cisco.com
```
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

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.

Before You Begin

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

- 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 password password-phrase
5. crypto pki import trustpointname pkcs12 source-url password password-phrase
6. exit
7. show crypto key mypubkey rsa

Detailed steps

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>crypto pki trustpoint name</td>
<td>Creates the trustpoint name that is to be associated with the RSA key pair and enters ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config)# crypto pki trustpoint my-ca</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>rsakeypair key-label [key-size [encryption-key-size]]</td>
<td>Specifies the key pair that is to be used with the trustpoint.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(ca-trustpoint)# rsakeypair my-keys</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>exit</td>
<td>Exits ca-trustpoint configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(ca-trustpoint)# exit</td>
<td></td>
</tr>
<tr>
<td>Command or Action</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>crypto pki export trustpointname pkcs12 destination-url password password-phrase</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config)# crypto pki export my-ca pkcs12 tftp://tftpserver/my-keys password mypassword123</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exports the RSA keys through the trustpoint name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The trustpointname argument enters the name of the trustpoint that issues the certificate that a user is going to export. When exporting the PKCS12 file, the trustpoint name is the RSA key name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The destination-url argument enters the file system location of the PKCS12 file to which a user wants to import the RSA key pair. See the crypto pki export pkcs12 password command page for more information.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The password-phrase argument must be entered to encrypt the PKCS12 file for export.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>crypto pki import trustpointname pkcs12 source-url password password-phrase</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config)# crypto pki import my-ca pkcs12 tftp://tftpserver/my-keys password mypassword123</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Imports the RSA keys to the target router.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The trustpointname argument enters the name of the trustpoint that issues the certificate that a user is going to export or import. When importing, the trustpoint becomes the RSA key name.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The source-url argument specifies the file system location of the PKCS12 file to which a user wants to export the RSA key pair. See the crypto pki import pkcs12 password command page for more information.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The password-phrase must be entered to undo encryption when the RSA keys are imported.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong></td>
<td>exit</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config)# exit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exits global configuration mode.</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong></td>
<td>show crypto key mypubkey rsa</td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Router# show crypto key mypubkey rsa</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Optional) Displays the RSA public keys of your router.</td>
<td></td>
</tr>
</tbody>
</table>

---

**Exporting and Importing RSA Keys in PEM-Formatted Files**

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

**Before You Begin**

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

---

In the table above, the **Step 4** section describes the process of exporting RSA keys using the `crypto pki export` command. It includes an example command and explains the purpose of the command, which is to export RSA keys through a trustpoint name. The command includes arguments such as `trustpointname`, `destination-url`, `password`, and `password-phrase`.

**Step 5** details the importation of RSA keys using the `crypto pki import` command. It offers an example command and explains the purpose of the command, which is to import RSA keys to a target router. The command includes arguments like `trustpointname`, `source-url`, `password`, and `password-phrase`.

**Step 6** provides the `exit` command to exit the global configuration mode.

**Step 7** uses the `show crypto key mypubkey rsa` command to display the RSA public keys of the router.

---

**Exporting and Importing RSA Keys in PEM-Formatted Files**

This section instructs users on how to export and import RSA keys in PEM-formatted files. It includes a brief overview of what is required and how to perform the task. The process involves generating an RSA key pair and then marking it "exportable" as specified in another task. The `crypto pki export` and `crypto pki import` commands are used for exporting and importing RSA keys, respectively. The `exit` command is used to exit the configuration mode, and the `show crypto key mypubkey rsa` command is used to display RSA public keys.
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.

Security threats, as well as the cryptographic technologies to help protect against them, are constantly changing. For more information about the latest Cisco cryptographic recommendations, see the Next Generation Encryption (NGE) white paper.

### SUMMARY STEPS

1. `crypto key generate rsa {usage-keys | general-keys} label key-label [exportable]
2. `crypto pki export trustpoint pem {terminal | url destination-url} {3des | des} password password-phrase
3. `crypto pki import trustpoint pem [check | exportable | usage-keys] {terminal | url source-url} password password-phrase
4. `exit
5. `show crypto key mypubkey rsa

### DETAILED STEPS

**Step 1**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| `crypto key generate rsa {usage-keys | general-keys} label key-label [exportable]` | Generates the RSA key pair. To use PEM files, the RSA key pair must be labeled exportable.

**Example:**

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

**Step 2**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| `crypto pki export trustpoint pem {terminal | url destination-url} {3des | des} password password-phrase` | Exports the certificates and RSA keys that are associated with a trustpoint in a PEM-formatted file.

- Enter the `trustpoint` name that is associated with the exported certificate and RSA key pair. The trustpoint name must match the name that was specified through the `crypto pki trustpoint` command.
- Use the `terminal` keyword to specify the certificate and RSA key pair that is displayed in PEM format on the console terminal.
- Use the `url` keyword and `destination-url` argument to specify the URL of the file system where your router should export the certificates and RSA key pair.

**Example:**

Router(config)# crypto pki export mycs pem url nvram: 3des password mypassword123
### Purpose

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>• (Optional) the <code>3des</code> keyword exports the trustpoint using the Triple Data Encryption Standard (3DES) encryption algorithm.</td>
<td></td>
</tr>
<tr>
<td>• (Optional) the <code>des</code> keyword exports the trustpoint using the DES encryption algorithm.</td>
<td></td>
</tr>
<tr>
<td>• Use the <code>password-phrase</code> argument to specify the encrypted password phrase that is used to encrypt the PEM file for import.</td>
<td></td>
</tr>
</tbody>
</table>

**Tip**
Be sure to keep the PEM file safe. For example, you may want to store it on another backup router.

### Step 3

**crypto pki import trustpoint pem**

```plaintext
[check | exportable | usage-keys]
{terminal | url source-url}
password password-phrase
```

**Example:**

```plaintext
Router(config)# crypto pki import
mycs2 pem url nvram: password
mypassword123
```

Imports certificates and RSA keys to a trustpoint from PEM-formatted files.

- Enter the `trustpoint` name that is associated with the imported certificate and RSA key pair. The trustpoint name must match the name that was specified through the `crypto pki trustpoint` command.

- (Optional) Use the `check` keyword to specify that an outdated certificate is not allowed.

- (Optional) Use the `exportable` keyword to specify that the imported RSA key pair can be exported again to another Cisco device such as a router.

- (Optional) Use the `usage-keys` argument to specify that two RSA special usage key pairs will be imported (that is, one encryption pair and one signature pair), instead of one general-purpose key pair.

- Use the `source-url` argument to specify the URL of the file system where your router should import the certificates and RSA key pairs.

- Use the `password-phrase` argument to specify the encrypted password phrase that is used to encrypt the PEM file for import.

**Note**
The password phrase 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.

**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:**

```plaintext
Router(config)# exit
```

Exits global configuration mode.

### Step 5

**show crypto key mypubkey rsa**

**Example:**

```plaintext
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.

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.

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.

Before You Begin
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, on page 4.”
• Optionally, you can authenticate and enroll each router with the CA server.

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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>crypto key encrypt [write] rsa [name key-name] passphrase passphrase&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Router(config)# crypto key encrypt write rsa name pki.example.com passphrase password</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td>exit&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Router(config)# exit</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td>show crypto key mypubkey rsa&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Router# show crypto key mypubkey rsa</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td>crypto key lock rsa name key-name passphrase passphrase&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Router# crypto key lock rsa name pki.example.com passphrase password</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td>show crypto key mypubkey rsa&lt;br&gt;<strong>Example:</strong>&lt;br&gt;Router# show crypto key mypubkey rsa</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1** `enable` | Enables privileged EXEC mode.  
  - Enter your password if prompted. |
| **Example:**  
  Router> enable | |
| **Step 2** `configure terminal` | Enters global configuration mode. |
| **Example:**  
  Router# configure terminal | |
| **Step 3** `crypto key zeroize rsa [key-pair-label]` | 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. |
| **Example:**  
  Router(config)# crypto key zeroize rsa fancy-keys | |
| **Step 4** `crypto key zeroize pubkey-chain [index]` | 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. |
| **Example:**  
  Router(config)# crypto key zeroize pubkey-chain | |
| **Step 5** `exit` | Exits global configuration mode. |
| **Example:**  
  Router(config)# exit | |
| **Step 6** `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:**  
  Router# show crypto key mypubkey rsa | |
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]: 2048
% Generating 2048 bit RSA keys ...[OK]
!
crypto pki trustpoint mynewtp
rsakeypair mykeys
exit
```

```
crypto pki export mytp pkcs12 flash:myexport password mypassword123
Destination filename [myexport]? Writing pkcs12 file to tftp:/mytftpserver/myexport
CRYPTO_PKI:Exported PKCS12 file successfully.
Verifying checksum... OK (0x3307)
!
July 8 17:30:09 GMT:%CRYPTO-6-PKCS12EXPORT_SUCCESS:PKCS #12 Successfully Exported.
```

**Router B**

```
crypto pki import mynewtp pkcs12 flash:myexport password mypassword123
Source filename [myexport]?
CRYPTO_PKI:Imported PKCS12 file successfully.
!
July 8 18:07:50 GMT:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
```
Exporting and Importing and RSA Keys in PEM Files Example

The following example shows the generation, exportation, and importation of the RSA key pair "mytp", and verifies its status:

! Generate the key pair
!
Router(config)# crypto key generate rsa general-purpose label mytp exportable

The name for the keys will be: mytp
How many bits in the modulus [512]: 2048
% Generating 2048 bit RSA keys ...[OK]
!
! Archive the key pair to a remote location, and use a good password.
!
Router(config)# crypto pki export mytp pem url nvram:mytp 3des password mypassword123

% Key name:mytp
Usage: General Purpose Key
Exporting public key...
Writing file to nvram:mytp.pub
Exporting private key...
Writing file to nvram:mytp.prv
!
! Import the key as a different name.
!
Router(config)# crypto pki import mytp2 pem url nvram:mytp2 password mypassword123

% Importing public key or certificate PEM file...
Source filename [mytp2.pub]?
Reading file from nvram:mytp2.pub
% Importing private key PEM file...
Source filename [mytp2.prv]?
Reading file from nvram:mytp2.prv% Key pair import succeeded.
!
!
Router# show crypto key mypubkey rsa
% Key pair was generated at:18:04:56 GMT Jun 6 2011
Key name:mycs
Usage: General Purpose Key
Key is exportable.
Key Data:
30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00E65253 9C30C12E 2959DF8E 7D4FA4D2 5517FB49 09453AB9 739A31CB AA6B1B8F 32F8F2E7 840997E AAD8CF8A 23C29E19 C458F4F0 53BB2FA5 48789F79 A1095115 75958BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7DB2CA51 4B7E9F79 A1095115 75958BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7DB2CA51 4B7E9F79 C9C96D2C 2F70B50D 3B4CDDAE F661041A 445AE11D 002EEF08 F2A627A0 5B020301 0001
% Key pair was generated at:18:17:25 GMT Jun 6 2011
Key name:mycs2
Usage: General Purpose Key
Key is not exportable.
Key Data:
30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00E65253 9C30C12E 2959DF8E 7D4FA4D2 5517FB49 09453AB9 739A31CB AA6B1B8F 32F8F2E7 840997E AAD8CF8A 23C29E19 C458F4F0 53BB2FA5 48789F79 A1095115 75958BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7DB2CA51 4B7E9F79 A1095115 75958BC3 5DFB5D7F BCF655BF 6317DB12 A8287795 7DB2CA51 4B7E9F79 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
% 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-----
MIICAzCCAcwIBAgIBATANBgkqhkiG9w0BAQUFADBOMQswCQYDVQQGEwJVUzES
waDeNOSI3WlDa0ANQ5DkVbkwgn0TqIIXJOCttjHnW6K1LMcMV5n
---------END CERTIFICATE-----
% Key name: aaa
Usage: General Purpose Key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type:4,ENCRYPTED
DEK-Info:DES-EDE3-CBC,ED68210B626BC81A
Urguv0jnjgQwMVQX2R5nbzzYH12vGLunpH/ixFeJu3JRVjbaAUpGk7VnPCT87
---------END RSA PRIVATE KEY------
```
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:

```plaintext
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.
```

Encrypting and Locking Private Keys on a Router Examples

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.

```plaintext
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 8AFE8340 0EC1E62B D40E7DCC
```
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
```

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

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of PKI, including RSA keys, certificate enrollment, and CAs</td>
<td>Cisco IOS PKI Overview: Understanding and Planning a PKI</td>
</tr>
</tbody>
</table>
Feature Information for RSA Keys Within a PKI

The following table provides release information about the feature or features described in this module. This table 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.
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to [www.cisco.com/go/cfn](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

**Table 1: Feature Information for RSA Keys Within a PKI**

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Software Releases</th>
<th>Feature Configuration Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS 4096-Bit Public Key Support</td>
<td>Cisco IOS XE Release 2.4</td>
<td>This feature introduces Cisco IOS 4096-bit peer public key support.</td>
</tr>
<tr>
<td>Exporting and Importing RSA Keys</td>
<td>Cisco IOS XE Release 2.1</td>
<td>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 commands were introduced or modified by this feature: <code>crypto ca export pkcs12</code>, <code>crypto ca import pkcs12</code>, <code>crypto key generate rsa (IKE)</code></td>
</tr>
<tr>
<td>Import of RSA Key Pair and Certificates in PEM Format</td>
<td>Cisco IOS XE Release 2.1</td>
<td>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 commands were introduced by this feature: <code>crypto ca export pem</code>, <code>crypto ca import pem</code>, <code>crypto key export pem</code>, <code>crypto key import pem</code></td>
</tr>
<tr>
<td>Multiple RSA Key Pair Support</td>
<td>Cisco IOS XE Release 2.1</td>
<td>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 commands were introduced or modified by this feature: <code>crypto key generate rsa</code>, <code>crypto key zeroize rsa</code>, <code>rsakeypair</code></td>
</tr>
<tr>
<td>Feature Name</td>
<td>Software Releases</td>
<td>Feature Configuration Information</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Protected Private Key Storage</td>
<td>Cisco IOS XE Release 2.1</td>
<td>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 commands were introduced or modified by this feature: <code>crypto key decrypt rsa</code>, <code>crypto key encrypt rsa</code>, <code>crypto key lock rsa</code>, <code>crypto key unlock rsa</code>, <code>show crypto key mypubkey rsa</code></td>
</tr>
</tbody>
</table>