



# Configuring Network Data Encryption

---

This chapter describes how to configure your router for network data encryption. This chapter includes the following sections:

- Why Encryption?
- Cisco's Implementation of Encryption
- Additional Sources of Information
- Pework: Before You Configure Encryption
- Configure Encryption
- Configure Encryption with GRE Tunnels
- Configure Encryption with an ESA in a VIP2
- Configure Encryption with an ESA in a Cisco 7200 Series Router
- Customize Encryption (Configure Options)
- Turn Off Encryption
- Testing and Troubleshooting Encryption
- Encryption Configuration Examples

---

**Note** Whenever the term “encryption” is used in this chapter, it refers only to encryption of network data, not to other types of encryption.

---

For a complete description of the encryption commands in this chapter, refer to the “Network Data Encryption Commands” chapter in the *Security Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

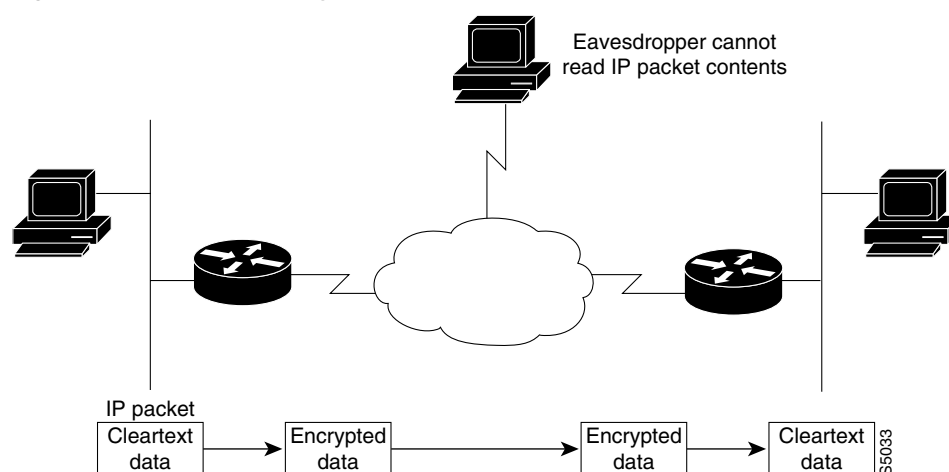
## Why Encryption?

Data that traverses unsecured networks is open to many types of attacks. Data can be read, altered, or forged by anybody who has access to the route that your data takes. For example, a protocol analyzer can read packets and gain classified information. Or, a hostile party can tamper with packets and cause damage by hindering, reducing, or preventing network communications within your organization.

Encryption provides a means to safeguard network data that travels from one Cisco router to another across unsecured networks. Encryption is particularly important if classified, confidential, or critical data is being sent.

Figure 9 illustrates the encryption of an IP packet as it travels across an unsecured network.

**Figure 9 IP Packet Encryption**



## Cisco's Implementation of Encryption

The following sections answer these questions:

- What Gets Encrypted?
- Where Are Packets Encrypted and Decrypted in the Network?
- When Can Encrypted Packets Be Exchanged?
- How Does an Encrypting Router Identify Other Peer Encrypting Routers?
- What Standards Are Implemented in Cisco's Encryption?
- How Does Cisco's Encryption Work?

### What Gets Encrypted?

Network data encryption is provided at the IP packet level—only IP packets can be encrypted. (If you wish to encrypt a network protocol other than IP, you must encapsulate the protocol within an IP packet.)

An IP packet is encrypted/decrypted only if the packet meets criteria you establish when you configure a router for encryption.

When encrypted, individual IP packets can be detected during transmission, but the IP packet contents (payload) cannot be read. Specifically, the IP header and upper-layer protocol headers (for example, TCP or UDP) are not encrypted, but all payload data within the TCP or UDP packet will be encrypted, and therefore not readable during transmission.

## Where Are Packets Encrypted and Decrypted in the Network?

The actual encryption and decryption of IP packets occur only at routers that you configure for network data encryption. Such routers are considered to be *peer encrypting routers* (or simply *peer routers*). Intermediate hops do not participate in encryption/decryption.

Often, peer routers are situated at the edges of unsecured networks (such as the Internet), in order to provide secure communications between two secured networks that are physically separated. Cleartext (not encrypted) traffic that enters a peer router from the secure network side is encrypted and forwarded across the unsecure network. When the encrypted traffic reaches the remote peer router, the router decrypts the traffic before forwarding it into the remote secure network.

Packets are encrypted at one peer router's outbound interface and decrypted at the other peer router's inbound interface.

## When Can Encrypted Packets Be Exchanged?

Encrypted packets can be exchanged between peer routers only during encrypted sessions. When a peer router detects a packet that should be encrypted, an encrypted session must first be established. After an encrypted session is established, encrypted traffic can pass freely between peer routers. When the session expires, a new session must be established before encrypted traffic can continue to be sent.

## How Does an Encrypting Router Identify Other Peer Encrypting Routers?

During the setup of every encrypted session, both participating peer routers attempt to authenticate each other. If either authentication fails, the encrypted session will not be established, and no encrypted traffic will pass. Peer authentication ensures that only known, trusted peer routers exchange encrypted traffic, and prevents routers from being tricked into sending sensitive encrypted traffic to illegitimate or fraudulent destination routers.

## What Standards Are Implemented in Cisco's Encryption?

To provide encryption services, Cisco implements the following standards: Digital Signature Standard (DSS), the Diffie-Hellman (DH) public key algorithm, and Data Encryption Standard (DES). DSS is used for peer router authentication. The DH algorithm and DES standard are used to initiate and conduct encrypted communication sessions between participating peer routers.

## How Does Cisco's Encryption Work?

The following sections provide an overview of Cisco's encryption process:

- You Enable Peer Router Authentication with a DSS Key Exchange
- A Router Establishes an Encrypted Session with a Peer
- Peer Routers Encrypt and Decrypt Data During an Encrypted Session

## You Enable Peer Router Authentication with a DSS Key Exchange

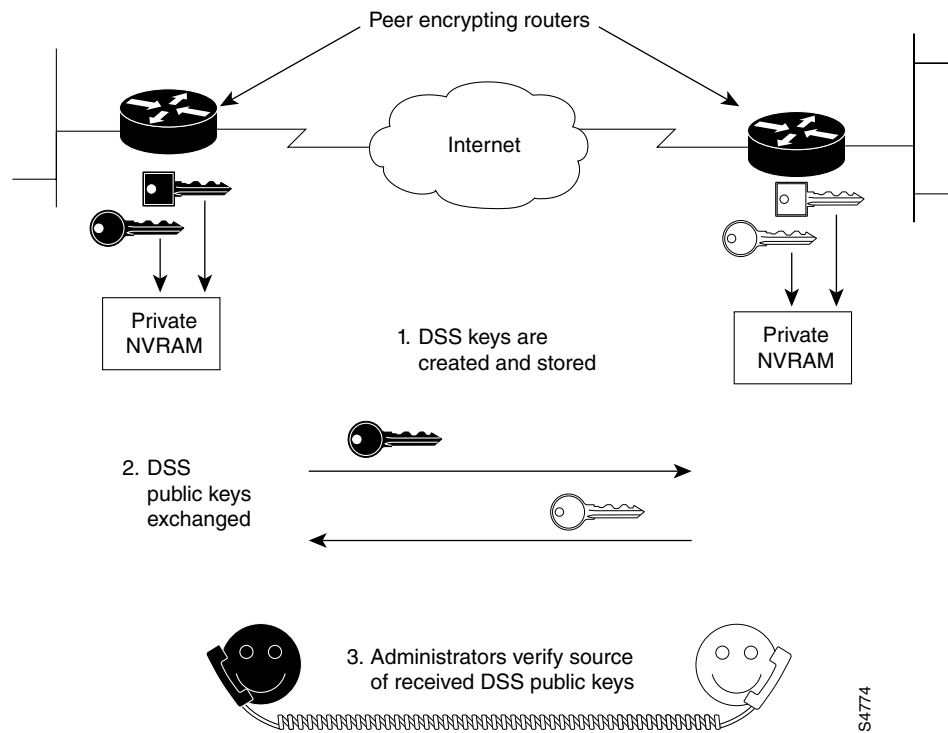
Peer router authentication occurs during the setup of each encrypted session. But before peer routers can authenticate each other, you must generate Digital Signature Standard (DSS) keys (both public and private DSS keys) for each peer, and you must exchange (and verify) the DSS public keys with each peer (see Figure 10). You generate and exchange DSS keys only once per peer, and afterwards these DSS keys will be used each time an encrypted session occurs. (Generating and exchanging DSS keys are described later in the section “Configure Encryption.”)

Each peer router's DSS keys are unique: a unique DSS public key, and a unique DSS private key. DSS public and private keys are stored in a private portion of the router's NVRAM, which cannot be viewed with commands such as **show configuration**, **show running-config**, or **write terminal**. If you have a router with an Encryption Service Adapter (ESA), DSS keys are stored in the tamper resistant memory of the ESA.

The DSS private key is not shared with any other device. However, the DSS public key is distributed to all other peer routers. You must cooperate with the peer router's administrator to exchange public keys between the two peer routers, and you and the other administrator must verbally verify to each other the public key of the other router. (The verbal verification is sometimes referred to as “voice authentication.”)

When an encrypted session is being established, each router uses the peer's DSS public key to authenticate the peer. The process of authenticating peers and establishing encrypted sessions is described next.

Figure 10 Exchanging DSS Keys (Overview)



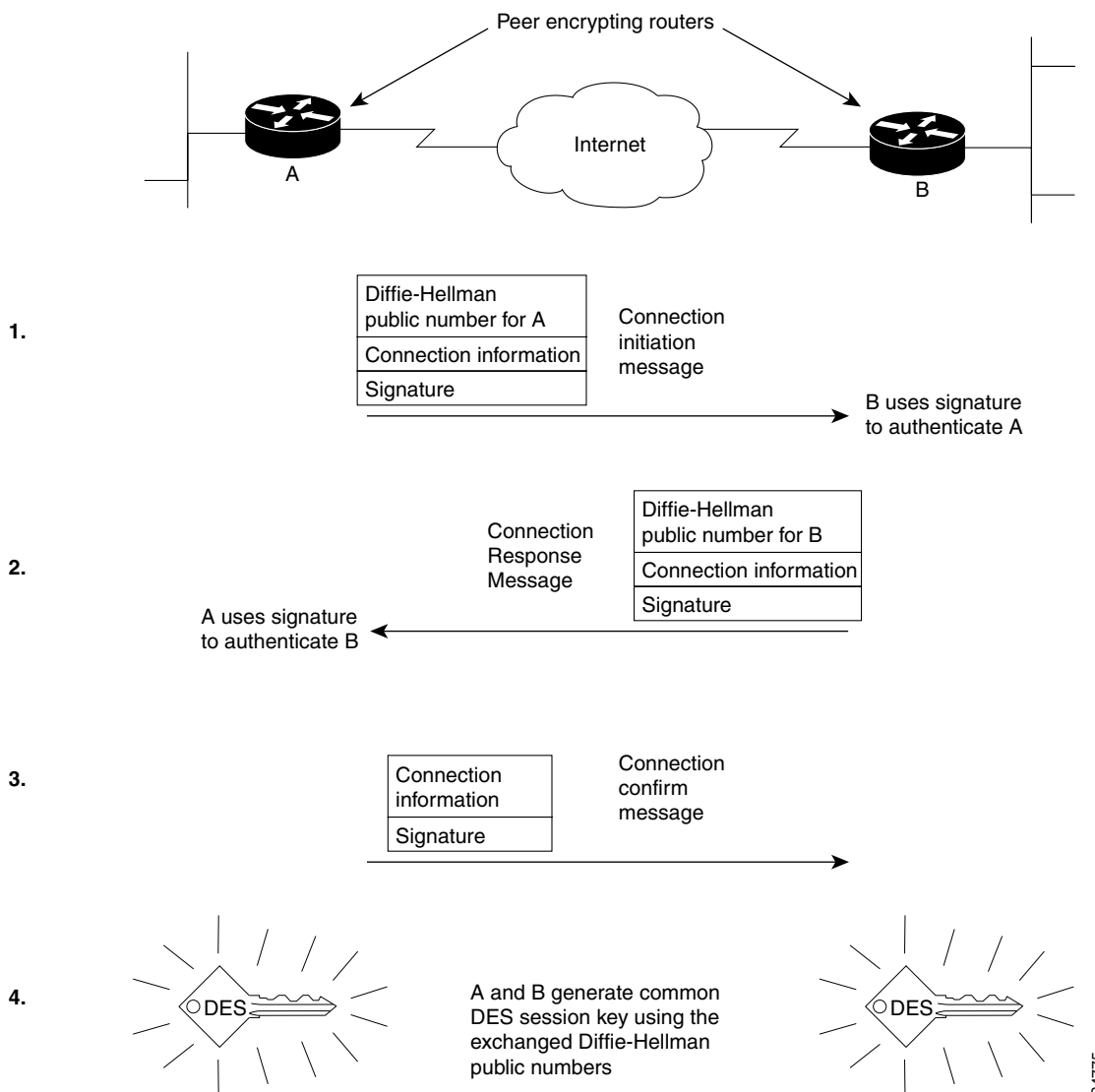
## A Router Establishes an Encrypted Session with a Peer

An encrypted session must be established before a Cisco router can send encrypted data to a peer router. (See Figure 11.) An encrypted session is established whenever a router detects an IP packet that should be encrypted and no encrypted session already exists.

To establish a session, two peer routers exchange connection messages. These messages have two purposes. The first purpose is to authenticate each router to the other. Authentication is accomplished by attaching “signatures” to the connection messages: A signature is a character string that is created by each local router using its own DSS private key, and verified by the other (remote) router using the local router’s DSS public key (previously exchanged). A signature is always unique to the sending router and cannot be forged by any other device. When a signature is verified, the router that sent the signature is authenticated.

The second purpose of the connection messages is to generate a temporary DES key (“session key”), which is the key that will be used to actually encrypt data during the encrypted session. To generate the DES key, Diffie-Hellman (DH) numbers must be exchanged in the connection messages. Then, the DH numbers are used to compute a common DES session key that is shared by both routers.

**Figure 11 Establishing an Encrypted Session**

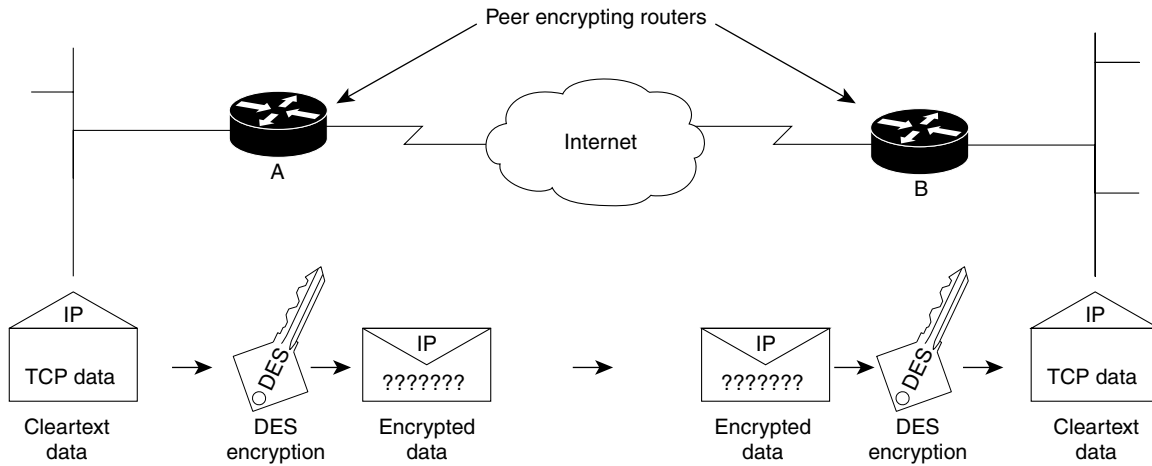


### Peer Routers Encrypt and Decrypt Data During an Encrypted Session

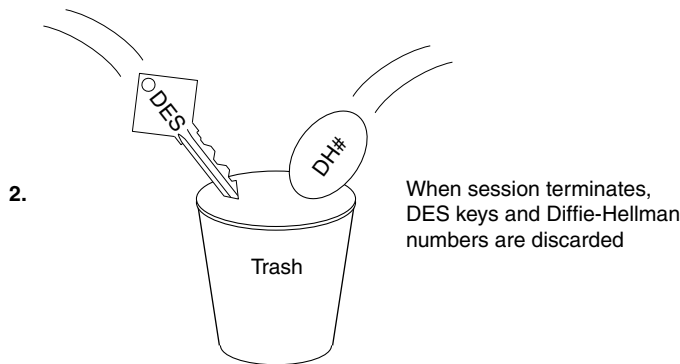
After both peer routers are authenticated and the session key (DES key) has been generated, data can be encrypted and transmitted. A DES encryption algorithm is used with the DES key to encrypt and decrypt IP packets during the encrypted session. (See Figure 12.)

An encrypted communication session will terminate when the session times out. When the session terminates, both the DH numbers and the DES key are discarded. When another encrypted session is required, new DH numbers and DES keys will be generated.

**Figure 12** Encrypting Data



1. DES key is used by routers A and B to encrypt outbound IP traffic and to decrypt inbound IP traffic



S4776

### Additional Sources of Information

The following reading material can provide additional background information about network data encryption, including theory, standards, and legal requirements.

*Applied Cryptography*, Bruce Schneier.

*Network Security: Private Communication in a Public World*, Kaufman, Perlman, and Specinen.

*Actually Useful Internet Security Techniques*, Larry J. Hughes, Jr.

*FIPS140*, Federal Information Processing Standard

Defense Trade Regulations (Parts 120 to 126)

*Information Security and Privacy in Network Environments*, Office of Technology Assessment

## Prework: Before You Configure Encryption

You should understand and follow the guidelines in this section *before* attempting to configure your system for network data encryption. This section describes the following guidelines:

- Identify Peer Routers
- Consider Your Network Topology
- Identify Crypto Engines within Each Peer Router
- Understand Implementation Issues and Limitations

### Identify Peer Routers

You must identify all peer routers that will be participating in encryption. Peer routers are routers configured for encryption, between which all encrypted traffic is passed. These peers are usually routers within your administrative control that will be passing IP packets over an uncontrolled network (such as the Internet). Participating peer routers might also include routers not within your administrative control; however, this should only be the case if you share a trusted, cooperative relationship with the other router's administrator. This person should be known and trusted by both you and your organization.

Peer routers should be located within a network topology per the guidelines listed next.

### Consider Your Network Topology

Take care in choosing a network topology between peer encrypting routers. Particularly, you should set up the network so that a stream of IP packets must use exactly one pair of encrypting routers at a time. Do not nest levels of encrypting routers. (That is, do not put encrypting routers in between two peer encrypting routers.)

Frequent route changes between pairs of peer encrypting routers, including for purposes of load balancing, will cause excessive numbers of connections to be set up and very few data packets to be delivered. Note that load balancing can still be used, but only if done transparently to the encrypting peer routers. That is, peer routers should not participate in the load balancing: only devices in between the peer routers should provide load balancing.

A common network topology used for encryption is a hub-and-spoke arrangement between an enterprise router and branch routers. Also, Internet firewall routers are often designated as peer encrypting routers.

### Identify Crypto Engines within Each Peer Router

Encryption is provided by a software service called a *crypto engine*. To perform encryption at a router, you must first configure the router's crypto engine to be an encrypting peer; then you can configure any interface governed by that crypto engine to perform encryption. (To configure a crypto engine, you must at a minimum generate and exchange DSS keys for that engine, as described later in the section "Configure Encryption.")

Depending on your hardware configuration, different crypto engines will govern different router interfaces. In some instances, you may even need to configure multiple crypto engines as peers within a single router, particularly if a router has multiple interfaces that you want to use for encryption, and those interfaces are governed by different crypto engines.

There are three types of crypto engines—the Cisco IOS crypto engine, the VIP2 crypto engine, and the ESA crypto engine.

If you have a Cisco 7200, RSP7000, or 7500 series router with one or more VIP2 boards (VIP2-40 or higher) or ESA cards, your router can have multiple crypto engines. All other routers have only one crypto engine, the Cisco IOS crypto engine.

When you configure a crypto engine on a Cisco 7200, RSP7000, or 7500 series router, you must identify which engine you are configuring by specifying the engine's chassis slot number when you enter the crypto commands.

The three different crypto engines are described next.

### The Cisco IOS Crypto Engine

Every router with Cisco IOS encryption software has a Cisco IOS crypto engine. For many Cisco routers, the Cisco IOS crypto engine is the only crypto engine available. The only exceptions are the Cisco 7200, RSP7000, and 7500 series routers, which can also have additional crypto engines as described in the next two sections.

If a router has no additional crypto engines, the Cisco IOS crypto engine governs all the router interfaces: you must configure the Cisco IOS crypto engine before you can configure any router interface for encryption.

The Cisco IOS crypto engine is identified by the chassis slot number of the Route Switch Processor (RSP). (For routers with no RSP, the Cisco IOS crypto engine is selected by default and does not need to be specifically identified during configuration.)

### The VIP2 Crypto Engine (Cisco RSP7000 and 7500 Series Routers Only)

Cisco RSP7000 and 7500 series routers with a second-generation Versatile Interface Processor (VIP2) (version VIP2-40 or greater) have two crypto engines: the Cisco IOS crypto engine and the VIP2 crypto engine.

The VIP2 crypto engine governs all the VIP2 interfaces. The Cisco IOS crypto engine governs all remaining router interfaces. (These rules assume there is no ESA installed in the VIP2. If the VIP2 has an installed ESA, the interfaces are governed differently, as explained in the next section.)

The VIP2 crypto engine is identified by the chassis slot number of the VIP2.

### The Encryption Service Adapter Crypto Engine (Cisco 7200, RSP7000, and 7500 Series Routers Only)

Cisco 7200, RSP7000, and 7500 series routers with an Encryption Service Adapter (ESA) have an ESA crypto engine.

#### Cisco 7200 Series Routers with an ESA

When a Cisco 7200 router has an active ESA, the ESA crypto engine—not the Cisco IOS crypto engine—governs all the router interfaces. (With an inactive ESA, the Cisco IOS crypto engine governs all the router interfaces. On the Cisco 7200, you can select which engine is active; only one engine is active at a time.)

The ESA plugs into the Cisco 7200 chassis, and the ESA crypto engine is identified by the ESA's chassis slot number.

### Cisco RSP7000 and 7500 Series Routers with an ESA

The ESA and an adjoining port adapter plug into a VIP2 board. The ESA crypto engine—not the VIP2 crypto engine—governs the adjoining VIP2 port interfaces. The Cisco IOS crypto engine governs all remaining interfaces.

In a Cisco RSP7000 or 7500 series router, the ESA crypto engine is identified by the chassis slot number of the VIP2.

## Understand Implementation Issues and Limitations

Please note the following issues and limitations of encryption, described in this section:

- Encapsulation
- Multicast of Encrypted Traffic
- IP Fragmentation
- Restrictions for Switching Types with the VIP2
- Number of Simultaneous Encrypted Sessions
- Performance Impacts

### Encapsulation

You can use any type of encapsulation with IP encryption, except as follows: If you have a second-generation Versatile Interface Processor (VIP2) with a serial interface, encryption will not work for traffic on the serial interface unless you use the Point-to-Point Protocol (PPP), High-Level Data Link Control (HDLC) protocol, or Frame Relay protocol. For example, you cannot use encryption if you have X.25 or SMDS configured for the serial interface of a VIP2.

### Multicast of Encrypted Traffic

Encrypted multicast is not supported.

### IP Fragmentation

IP fragmentation is supported with encryption for all platforms except the VIP2. If you configure encryption for VIP2 interfaces, all IP fragments will be dropped.

### Restrictions for Switching Types with the VIP2

If you configure encryption for VIP2 interfaces on a Cisco RSP7000 or 7500 series router, you must use distributed switching (DSW) on the source and destination encrypting/decrypting interfaces.

This restriction means that any protocol that is not compatible with DSW, such as SMDS, cannot be used on VIP2 encrypting interfaces.

### Number of Simultaneous Encrypted Sessions

Each encrypting router can set up encrypted sessions with many other routers, if these are peer encrypting routers. Encrypting routers can also set up multiple simultaneous encrypted sessions with multiple peer routers. Up to 299 concurrent encrypted sessions per router can be supported.

### Performance Impacts

Because of the high amount of processing required for encryption, if you use encryption heavily there will be performance impacts such as interface congestion or slowed CPU functioning. Using an ESA crypto engine rather than the Cisco IOS crypto engine can improve overall router performance because the Cisco IOS software will not be impacted by encryption processing.

## Configure Encryption

To pass encrypted traffic between two routers, you must configure encryption at both routers. This section describes the tasks required to configure encryption on one router: you must repeat these tasks for each peer encrypting router (routers that will participate in encryption).

To configure encryption on a router, complete the tasks described in the following sections:

- 1 Generate DSS Public/Private Keys (required to configure a crypto engine)
- 2 Exchange DSS Public Keys (required to configure a crypto engine)
- 3 Enable DES Encryption Algorithms (required to configure the router)
- 4 Define Crypto Maps and Assign Them to Interfaces (required to configure router interfaces)
- 5 Back Up Your Configuration

---

**Note** There are additional steps required if you configure encryption with GRE tunnels or if you configure encryption with a Data Encryption Service Adaptor (ESA). These additional steps are described later in this chapter, in the sections “Configure Encryption with GRE Tunnels,” “Configure Encryption with an ESA in a VIP2,” and “Configure Encryption with an ESA in a Cisco 7200 Series Router.” Before you configure encryption, refer to these other sections as appropriate.

---

For examples of the configuration in this section, see the section “Encryption Configuration Examples” at the end of this chapter.

### Generate DSS Public/Private Keys

You must generate DSS keys for each crypto engine you will use. If you will use more than one crypto engine, you must generate DSS keys separately for each engine. (These are the crypto engines you previously identified per the description in the earlier section “Identify Crypto Engines within Each Peer Router.”)

The DSS key pair that you generate is used by peer routers to authenticate each other before each encrypted session. The same DSS key pair is used by a crypto engine with all its encrypted sessions (regardless of the peer encrypting router that it connects to).

To generate DSS keys for a crypto engine, perform at least the first of the following tasks, in global configuration mode:

Task	Command
Generate DSS public and private keys.	<b>crypto gen-signature-keys</b> <i>key-name</i> [ <i>slot</i> ]
View your DSS public key (private key not viewable).	<b>show crypto mypubkey</b> [ <i>slot</i> ]
Save DSS keys to private NVRAM. (Complete this task only for Cisco IOS crypto engines.)	<b>copy running-config startup-config</b>

---

**Note** You must perform the **copy running-config startup-config** (previously **write memory**) command to save Cisco IOS crypto engine DSS keys to a private portion of NVRAM. DSS keys are *not* saved with your configuration when you perform a **copy running-config rcp** or **copy running-config tftp** (previously **write network**) command.

---

If you are generating keys for an ESA crypto engine, the following occurs during DSS key generation:

- You are prompted to enter a password.
  - If you previously used the **crypto zeroize** command to reset the ESA, you should create a new password for the ESA at this time.
  - If you previously used the **crypto clear-latch** command to reset the ESA, you should now use the password you assigned when you reset the ESA. If you do not remember the password, you must clear the ESA with the **crypto zeroize** command; you can then generate keys and create a new password for the ESA.
- The DSS keys are automatically saved to the tamper resistant memory of the ESA.

Configuring encryption with an ESA is described later in the sections “Configure Encryption with an ESA in a VIP2” and “Configure Encryption with an ESA in a Cisco 7200 Series Router.”

## Exchange DSS Public Keys

You must exchange DSS public keys with all participating peer routers. This allows peer routers to authenticate each other at the start of encrypted communication sessions.

If your network contains several peer encrypting routers, you need to exchange DSS keys multiple times (once for each peer router). If you ever add an encrypting peer router to your network topology, you will then need to exchange DSS keys with the new router to enable encryption to occur with that new router.

---

**Note** When you exchange DSS keys, you must make a phone call to the administrator of the peer encrypting router. You need to be in voice contact with the other administrator during the key exchange in order to voice authenticate the source of exchanged DSS public keys.

---

You must exchange the DSS public keys of each crypto engine that you will use.

To successfully exchange DSS public keys, you must cooperate with a trusted administrator of the other peer router. You and the administrator of the peer router must complete the following steps in the order given (refer to Figure 13):

- Step 1** Call the other administrator on the phone. Remain on the phone with this person until you complete all the steps in this list.
- Step 2** You and the other administrator decide which of you will be called “PASSIVE” and which will be called “ACTIVE.”

**Step 3** PASSIVE enables a DSS exchange connection by performing the following task in global configuration mode:

Task	Command
Enable a DSS exchange connection.	<b>crypto key-exchange passive</b> [ <i>tcp-port</i> ]

**Step 4** ACTIVE initiates a DSS exchange connection and sends a DSS public key by performing the following task in global configuration mode:

Task	Command
Initiate connection and send DSS public key.	<b>crypto key-exchange</b> <i>ip-address key-name</i> [ <i>tcp-port</i> ]

The serial number and Fingerprint of ACTIVE's DSS public key will display on both of your screens. The serial number and Fingerprint are numeric values generated from ACTIVE's DSS public key.

**Step 5** You both verbally verify that the serial number is the same on both your screens, and that the Fingerprint is the same on both your screens.

**Step 6** If the displayed serial numbers and Fingerprints match, PASSIVE should agree to accept ACTIVE's DSS key by typing **y** at the prompt.

**Step 7** PASSIVE sends ACTIVE a DSS public key by pressing <Return> at the screen prompt and selecting a crypto engine at the next prompt.

**Step 8** PASSIVE's DSS serial number and Fingerprint display on both of your screens.

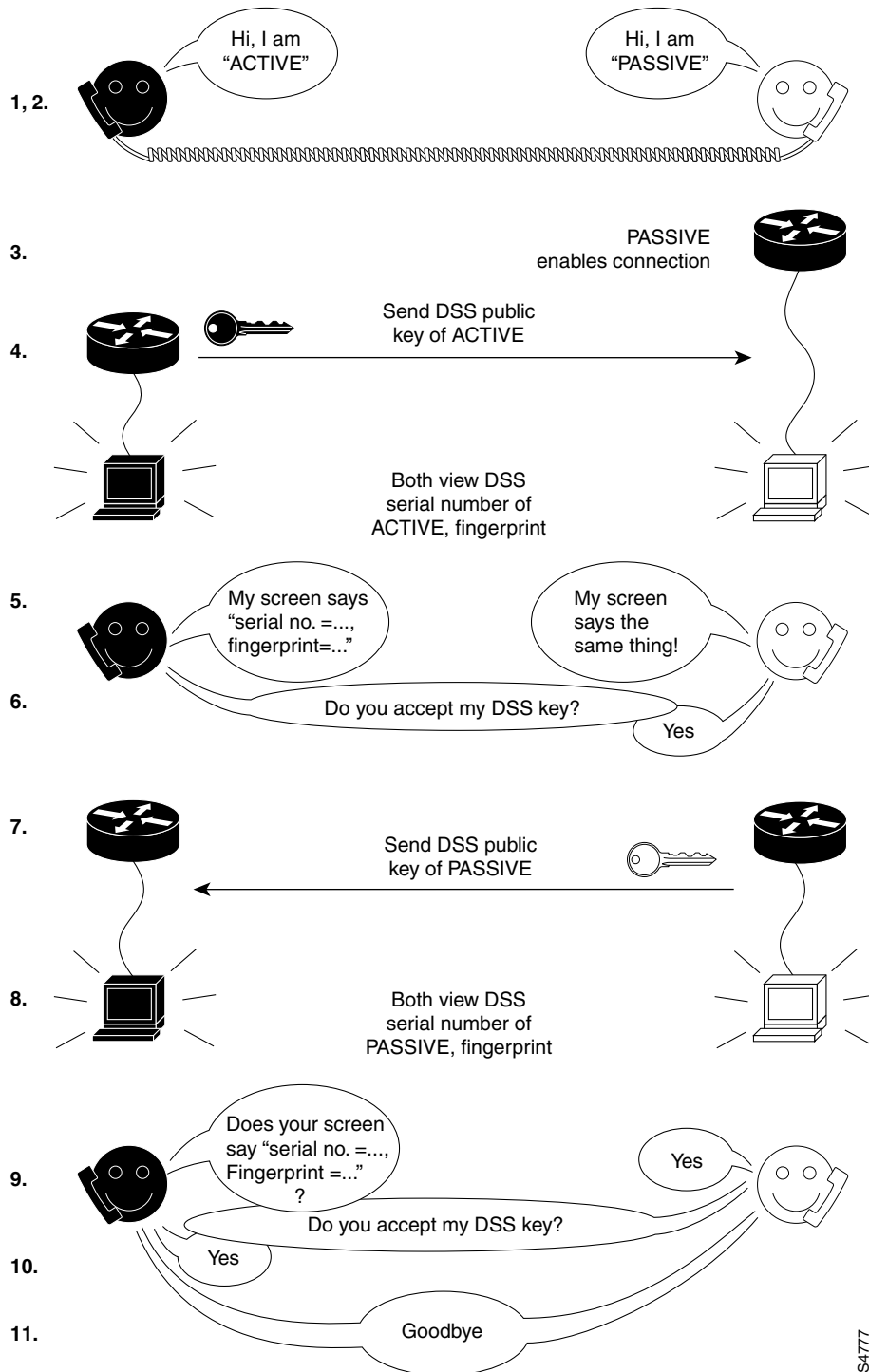
**Step 9** As before, you both verbally verify that the PASSIVE's DSS serial number and Fingerprint match on your two screens.

**Step 10** ACTIVE agrees to accept PASSIVE's DSS public key.

**Step 11** The exchange is complete, and you can end the phone call.

The previous steps (illustrated in Figure 13) must be accomplished between your router and a peer router for every peer router with which you will be conducting encrypted sessions.

Figure 13 Exchange DSS Public Keys



S4777

### Enable DES Encryption Algorithms

Cisco routers use Data Encryption Standard (DES) encryption algorithms and DES keys to encrypt and decrypt data. You must globally enable (turn on) all the DES encryption algorithms that your router will use during encrypted sessions. If a DES algorithm is not enabled globally, you will not be able to use it. (Enabling a DES algorithm once allows it to be used by all crypto engines of a router.)

To conduct an encrypted session with a peer router, you must enable at least one DES algorithm that the peer router also has enabled. You must configure the same DES algorithm on both peer routers for encryption to work.

Cisco supports the following four types of DES encryption algorithms:

- DES with 8-bit Cipher FeedBack (CFB)
- DES with 64-bit CFB
- 40-bit variation of DES with 8-bit CFB
- 40-bit variation of DES with 64-bit CFB

The 40-bit variations use a 40-bit DES key, which is easier for attackers to “crack” than basic DES which uses a 56-bit DES key. However, some international applications might require you to use 40-bit DES because of export laws. Also, 8-bit CFB is more commonly used than 64-bit CFB, but requires more CPU time to process. Other conditions might also exist that require you to use one or another type of DES.

---

**Note** If you are running an exportable image, you can only enable and use 40-bit variations of DES. You cannot enable or use the basic DES algorithms, which are not available with exportable images.

---

One DES algorithm is enabled for your router by default. If you do not plan to use the default DES algorithm, you may choose to disable it. If you are running a non-exportable image, the DES default algorithm will be basic DES with 64-bit CFB. If you are running an exportable image, the DES default algorithm will be the 40-bit variation of DES with 64-bit CFB.

If you do not know if your image is exportable or non-exportable, you can perform the **show crypto algorithms** command to determine which DES algorithms are currently enabled.

To globally enable one or more DES algorithms, perform one or more of the following tasks, in global configuration mode:

Task	Command
Enable DES with 8-bit or 64-bit CFB.	<b>crypto algorithm des [cfb-8   cfb-64]</b>
Enable 40-bit DES with 8-bit or 64-bit CFB.	<b>crypto algorithm 40-bit-des [cfb-8   cfb-64]</b>
View all enabled DES algorithms.	<b>show crypto algorithms</b>

## Define Crypto Maps and Assign Them to Interfaces

The purpose of this task is to tell your router which interfaces should encrypt/decrypt traffic, which IP packets to encrypt or decrypt at those interfaces, and also which DES encryption algorithm to use when encrypting/decrypting the packets.

There are actually three steps required to complete this task:

- Step 1** Set Up Encryption Access List (to be used in the crypto map definition)
- Step 2** Define Crypto Maps
- Step 3** Apply Crypto Maps to Interfaces

---

**Note** You should select which interfaces to configure so that traffic is encrypted at the outbound interface of the local peer router, and traffic is decrypted at the input interface of the remote peer.

---

### Set Up Encryption Access List

Encryption access lists are used in this step to define which IP packets will be encrypted and which IP packets will not be encrypted. Encryption access lists are defined using extended IP access lists. (Normally, IP access lists are used to filter traffic. Encryption access lists are *not* used to filter traffic but are used to specify which packets to encrypt or not encrypt.)

To set up encryption access lists for IP packet encryption, perform the following task in global configuration mode:

Task	Command
Specify conditions to determine which IP packets will be encrypted. (Enable or disable encryption for traffic that matches these conditions.) <sup>1</sup>	<pre><b>access-list</b> <i>access-list-number</i> [<b>dynamic</b> <i>dynamic-name</i> [<b>timeout</b> <i>minutes</i>]] {<b>deny</b>   <b>permit</b>} <i>protocol source source-wildcard destination destination-wildcard</i> [<b>precedence</b> <i>precedence</i>] [<b>tos</b> <i>tos</i>] [<b>log</b>]</pre> <p>or</p> <pre><b>ip access-list extended</b> <i>name</i></pre> <p>Follow with <b>permit</b> and <b>deny</b> statements as appropriate.</p>

1. You specify conditions using an IP access list designated by either a number or a name. The **access-list** command designates a numbered access list; the **ip access-list extended** command designates a named access list.

Using the **permit** keyword will cause all traffic that is passed between the specified source and destination addresses to be encrypted/decrypted by peer routers. Using the **deny** keyword prevents that traffic from being encrypted/decrypted by peer routers.

The encryption access list you define at the local router must have a “mirror-image” encryption access list defined at the remote router, so that traffic that is encrypted locally is decrypted at the remote peer.

See the *Security Command Reference* for complete details about the extended IP access list commands.

The encryption access list you define will be applied to an interface as an outbound encryption access list after you define a crypto map and apply the crypto map to the interface. (These two tasks are described in the next sections.)



**Caution** When you create encryption access lists, Cisco recommends *against* using the **any** keyword to specify source or destination addresses. Using the **any** keyword could cause extreme problems if a packet enters your router and is destined for a router that is not configured for encryption. This would cause your router to attempt to set up an encryption session with a nonencrypting router.

**Note** If your encryption access list defines more than 100 distinct source addresses or more than 10 destination addresses for a given source address, you need to change certain defaults as described later in the section “Change Encryption Access List Limits.”

**Note** If you view your router’s access lists by using a command such as **show ip access-lists**, *all* extended IP access lists will be shown in the command output. This includes extended IP access lists that are used for traffic filtering purposes as well as those that are used for encryption. The **show** command output does not differentiate between the two uses of the extended access lists.

## Define Crypto Maps

You must define exactly one crypto map for each physical interface that will send encrypted data to a peer encrypting router.

Crypto maps are used to specify which DES encryption algorithm(s) will be used in conjunction with each access list defined in the previous step. Crypto maps are also used to identify which peer routers will provide the remote end encryption services.

To define a crypto map, perform the following tasks. The first task is performed in global configuration mode; the other tasks are performed in crypto map configuration mode.

Task	Command
Name the crypto map. (Executing this command causes you to enter the crypto map configuration mode.)	<b>crypto map</b> <i>map-name seq-num</i>
Specify the remote peer router.	<b>set peer</b> <i>key-name</i>
Specify at least one encryption access list.	<b>match address</b> [ <i>access-list-number</i>   <i>name</i> ]
Specify at least one DES encryption algorithm. (This must be an algorithm you previously enabled.)	<b>set algorithm des</b> [ <b>cfb-8</b>   <b>cfb-64</b> ] or <b>set algorithm 40-bit-des</b> [ <b>cfb-8</b>   <b>cfb-64</b> ]

**Note** If you are running an exportable image, you can only specify 40-bit variations of DES. You cannot enable or use the basic DES algorithms, which are not available with exportable images.

To define an additional, different set of parameters for the same interface, repeat the steps in the previous task list, using the same *map-name* but use a different *seq-num* for the crypto map command. For more information about this, refer to the **crypto map** command description in the “Network Data Encryption Commands” chapter of the *Security Command Reference*.

## Apply Crypto Maps to Interfaces

This step puts into effect the crypto maps just defined. You must apply exactly one crypto map to each physical interface that will encrypt outbound data and decrypt inbound data. This interface provides the encrypted connection to a peer encrypting router. An interface will not encrypt/decrypt data until you apply a crypto map to the interface.

To apply a crypto map to an interface, perform the following task in interface configuration mode:

Task	Command
Apply a crypto map to an interface.	<b>crypto map</b> <i>map-name</i>

## Back Up Your Configuration

Cisco recommends that after you configure your router for encryption, you make a backup of your configuration. (Be careful to restrict unauthorized access of this backed-up configuration.)

You can learn how to back up your configuration in the “Modifying, Downloading, and Maintaining Configuration Files” chapter of the *Configuration Fundamentals Configuration Guide*.

## Configure Encryption with GRE Tunnels

When GRE tunnel endpoints are located at the peer encrypting routers, you can configure encryption so that all traffic through the GRE tunnel is encrypted.

Note that you cannot selectively encrypt GRE tunnel traffic: either all the GRE tunnel traffic is encrypted, or no GRE tunnel traffic is encrypted.

To configure encryption with GRE tunnels, perform the same basic tasks described previously in the section “Configure Encryption.” However, while you complete basic task 4, Define Crypto Maps and Assign Them to Interfaces, you also must follow the additional instructions described next (for two cases):

- Encrypt Only GRE Tunnel Traffic
- Encrypt GRE Tunnel Traffic and Other Traffic

For examples of configuring encryption with a GRE tunnel, see the section “Examples of Configuring Encryption With GRE Tunnels,” later in this chapter.

### Encrypt Only GRE Tunnel Traffic

To encrypt only traffic through the GRE tunnel, follow these two additional instructions:

- When you set up your encryption access list, the list should contain only one criteria statement. In this one statement, specify **gre** as the protocol, specify the tunnel source address as the source, and specify the tunnel destination address as the destination.
- Apply the crypto map to both the physical interface and to the tunnel interface. (Without GRE tunnels, you only had to apply the crypto map to the physical interface.)

Remember to apply a crypto map to the physical interface and tunnel interface at both ends of the GRE tunnel.

### Encrypt GRE Tunnel Traffic and Other Traffic

To encrypt both GRE tunnel traffic and other specified non-GRE tunnel traffic, follow these three additional instructions:

- Create two separate encryption access lists as follows:
    - The first encryption access list should contain only one criteria statement. In this one statement, specify **gre** as the protocol, specify the tunnel source address as the source, and specify the tunnel destination address as the destination.
    - In the second encryption access list, specify which non-GRE traffic should be encrypted. (For example, you could specify **tcp** as a protocol, and specify a subnet source/wildcard and a subnet destination/wildcard.)
  - Create two separate crypto maps as follows:
    - In the first crypto map, specify the first encryption access list, along with a DES algorithm and the remote peer.
    - In the second crypto map, create at least two separate subdefinitions. The first subdefinition should exactly match the statements in the first crypto map. The second subdefinition should specify the second encryption access list, a DES algorithm, and the remote peer.
  - Apply the first crypto map to tunnel interface, and apply the second crypto map to the physical interface. (Without GRE tunnels, you only had to apply one crypto map to the physical interface.)
- Remember to apply a crypto map to the physical interface and tunnel interface at both ends of the GRE tunnel.

### Configure Encryption with an ESA in a VIP2

To configure encryption with a Data Encryption Service Adaptor (ESA), there are additional instructions that you must follow, in addition to the basic encryption configuration tasks described previously in the section, “Configure Encryption.”

Before you start to accomplish the basic tasks, be sure to read the information in this section.

This section describes configuration for an ESA plugged into a VIP2 on a Cisco RSP7000 or 7500 series router.

To configure encryption with an ESA plugged into a VIP2, complete these tasks in this order:

- 1 Reset the ESA
- 2 Perform Additional Encryption Configuration

---

**Note** If you ever remove and reinstall the ESA or the VIP2, you must reset the ESA again.

---

For examples of ESA-specific configuration tasks, see the section “Examples of ESA-Specific Encryption Configuration Tasks,” later in this chapter.

### Reset the ESA

If you do not reset the ESA in a VIP2, the ESA crypto engine will not be used; instead, the VIP2 crypto engine will govern all the VIP2 interfaces (and the Cisco IOS crypto engine will govern the other router interfaces).

Before an ESA is reset, the ESA's "Tampered" LED is lit.

To reset an ESA, complete one of the following bulleted tasks:

- Reset an ESA that has never been used before. Complete this task in global configuration mode:

Task	Command
Reset the ESA by clearing the ESA hardware latch.	<b>crypto clear-latch slot</b>
When prompted, create a new password for the ESA.	<i>password</i>

- Reset an ESA that was previously used, and you know the ESA password. Complete this task in global configuration mode:

Task	Command
Reset the ESA by clearing the ESA hardware latch.	<b>crypto clear-latch slot</b>
When prompted, type the ESA password previously assigned.	<i>password</i>

- Reset an ESA that was previously used, but you do not know the ESA password. Complete this task in global configuration mode:

Task	Command
Clear the ESA. (This deletes all DSS keys for the ESA.)	<b>crypto zeroize slot</b>

## Perform Additional Encryption Configuration

After you reset the ESA in a VIP2, continue configuring encryption by following the instructions in one of the following bullets.

- If the router, VIP2 and ESA were never previously configured for encryption, complete all the tasks described earlier in the section "Configure Encryption."
- If the ESA was never previously configured for encryption, but the router and VIP2 interfaces were configured for encryption, complete only the following two tasks, described earlier in the section "Configure Encryption":
  - Generate DSS Public/Private Keys (for the ESA crypto engine)
  - Exchange DSS Public Keys

---

**Note** When you generate DSS keys for the ESA, you could give the keys the same *key-name* as you previously gave the VIP2 keys. If you do not give the keys the same *key-name*, you need to make sure that all remote peer routers update their crypto maps to include the new name of the ESA crypto engine as their peer.

---

As always, remember to back up your configuration when you are done.

- If the router, VIP2, and ESA were all previously configured for encryption, you might not need to complete any additional configuration. However, you will need additional configuration in at least the following cases (see the section “Configure Encryption” for descriptions of the tasks):
  - If you have any concern that the old ESA keys are compromised, you should regenerate and exchange new DSS keys for the ESA. (Use the same ESA *key-name* previously assigned.)
  - If the ESA was relocated and now governs different interfaces than before, either all peer routers must update their crypto maps to reflect the changed peers, or you must regenerate and exchange new DSS keys for the ESA, assigning the *key-name* that is currently in the peer routers’ crypto maps.
  - If you previously reset the ESA with the **crypto zeroize** command because you did not know the ESA password, you must at a minimum generate and exchange DSS keys for the ESA crypto engine.

As always, remember to back up your configuration when you are done.

## Configure Encryption with an ESA in a Cisco 7200 Series Router

To configure encryption with a Data Encryption Service Adaptor (ESA), there are additional instructions that you must follow in addition to the basic encryption configuration tasks described previously in the section “Configure Encryption.”

Before you start to accomplish the basic tasks, be sure to read the information in this section.

This section describes configuration for an ESA plugged into a Cisco 7200 series router.

For examples of ESA-specific configuration tasks, see the section “Examples of ESA-Specific Encryption Configuration Tasks,” later in this chapter.

### Required Tasks

Complete the following tasks in this order (see following sections for descriptions):

- 1 Reset the ESA
- 2 Perform Additional Encryption Configuration
- 3 Enable the ESA

---

**Note** If you ever remove and reinstall the ESA, you must reset the ESA again and re-enable the ESA.

---

### Optional Tasks

You can optionally complete these additional tasks (see following sections for descriptions):

- Select a Crypto Engine (After encryption is configured, you might want to change which crypto engine to use—the Cisco IOS crypto engine or the ESA crypto engine.)
- Delete DSS Keys (If you ever remove or relocate the ESA or the Cisco 7200, you might want to delete DSS keys, to reduce any potential security risk.)

## Reset the ESA

Before an ESA is reset, the ESA's "Tampered" LED is lit.

To reset an ESA in a Cisco 7200 series router, complete one of the following bulleted tasks:

- Reset an ESA that has never been used before. Complete this task in global configuration mode:

Task	Command
Reset the ESA by clearing the ESA hardware latch.	<b>crypto clear-latch</b> <i>slot</i>
When prompted, create a new password for the ESA.	<i>password</i>

- Reset a previously used ESA when:
  - You know the ESA password.
  - You know or suspect that either the ESA, router, or router interfaces need additional configuration. (For example, the ESA's previous configuration was not complete or is uncertain; or you know you want to generate new DSS keys for the ESA; or the router is not configured for encryption.)

If you meet all the criteria just described, reset the ESA, in global configuration mode:

Task	Command
Reset the ESA by clearing the ESA hardware latch.	<b>crypto clear-latch</b> <i>slot</i>
When prompted, type the ESA password previously assigned.	<i>password</i>
If prompted to enable the ESA, type <b>no</b> .	<b>no</b>

- Reset a previously used ESA when:
  - You know the ESA password.
  - The router has been configured for encryption (and may have already been encrypting traffic using the Cisco IOS crypto engine).
  - The ESA has already been configured for encryption with the same *key-name* as the Cisco IOS crypto engine and with DSS keys generated and exchanged, and you do not want to generate new keys.
  - All encryption configuration is already complete as described previously in the section "Configure Encryption," and you are ready to start encrypting traffic using the ESA crypto engine.

If you meet all the criteria just described, reset the ESA, in global configuration mode:

Task	Command
Reset the ESA by clearing the ESA hardware latch.	<b>crypto clear-latch</b> <i>slot</i>
When prompted, type the ESA password previously assigned.	<i>password</i>
When prompted to enable the ESA, type <b>yes</b> .	<b>yes</b>

---

**Note** After you reset the ESA as just described, the ESA will automatically become active and begin encrypting traffic. For this case only, you do not need to complete any additional encryption configuration. (But as always, be sure to back up your configuration.)

---

- Reset a previously used ESA when you do not know the ESA password. Complete this task in global configuration mode:

---

Task	Command
Clear the ESA. (This deletes all DSS keys for the ESA.)	<b>crypto zeroize slot</b>

---

### Perform Additional Encryption Configuration

After you reset the ESA in a Cisco 7200 series router, continue configuring encryption by following the instructions in one of the following bullets:

- If the router and ESA were never previously configured for encryption, complete all the tasks described earlier in the section “Configure Encryption,” then enable the ESA as described in the next section, “Enable the ESA.”
- If the ESA was never previously configured for encryption, but the router is configured for encryption, complete only the following two tasks described earlier in the section “Configure Encryption”:
  - Generate DSS Public/Private Keys (for the ESA crypto engine)
  - Exchange DSS Public Keys (for the ESA crypto engine)

---

**Note** When you generate DSS keys for the ESA, give the keys the same *key-name* previously assigned to the Cisco IOS crypto engine keys.

---

After you generate and exchange DSS keys for the ESA crypto engine, enable the ESA as described in the next section, “Enable the ESA.”

- If the router and ESA are both already configured for encryption, you might only need to enable the ESA as described in the next section, “Enable the ESA.”

However, in at least the following cases you will need additional configuration before you enable the ESA (see the section “Configure Encryption” for descriptions of the tasks):

- If the ESA has DSS keys generated but not exchanged with the peer routers, you must exchange the keys.
- If you have any concern that the ESA’s DSS keys are compromised, you should regenerate and exchange new DSS keys for the ESA, using the same *key-name* assigned to the router DSS keys.
- If the ESA was relocated from a different router, regenerate and exchange DSS keys, using the same *key-name* assigned to the router DSS keys.
- If you previously reset the ESA with the **crypto zeroize** command because you did not know the ESA password, you must at a minimum generate and exchange DSS keys for the ESA crypto engine.

## Enable the ESA

To enable an ESA in a Cisco 7200 series router, complete the following task in global configuration mode:

Task	Command
Enable the ESA.	<code>crypto esa enable slot</code>

**Note** If the Cisco IOS crypto engine is currently encrypting traffic when you enable the ESA, the session will be torn down, and a new session will be established using the ESA crypto engine. This could cause a momentary delay for encrypted traffic.

As always, remember to back up your configuration when you are done.

## Select a Crypto Engine

This is an optional task.

After encryption is configured on a Cisco 7200 series router with an ESA, you might want to change which crypto engine to use—the Cisco IOS crypto engine or the ESA crypto engine. This section describes how to switch from one crypto engine to the other.

You should only select a crypto engine if the engine is fully configured for encryption.

If you boot the router with an operational ESA installed, the ESA will be the active crypto engine upon bootup, by default. Otherwise, the Cisco IOS crypto engine will be the default active crypto engine.

**Note** If any encryption session is in progress when you switch from one crypto engine to the other, the session will be torn down, and a new session will be established using the newly selected crypto engine. This could cause a momentary delay for encrypted traffic.

## Select the Cisco IOS Crypto Engine

If the ESA crypto engine is encrypting traffic, but you want to cause the Cisco IOS crypto engine to encrypt the traffic instead, you can switch to the Cisco IOS crypto engine without removing the ESA. (You might want to do this for testing purposes.)



**Caution** Before you switch to the Cisco IOS crypto engine, be sure that the Cisco IOS crypto engine is configured with DSS keys generated and exchanged, and be sure that the DSS keys have the same *key-name* for both engines; otherwise, you will lose encryption capability when you switch engines.

To select the Cisco IOS crypto engine, perform the following task in global configuration mode:

Task	Command
Shut down the ESA.	<code>crypto esa shutdown slot</code>

After you select the Cisco IOS crypto engine, the Cisco IOS crypto engine will be the active engine, governing the router interfaces. The Cisco IOS crypto engine will perform the encryption services, and the ESA will be inactive.

### Select the ESA Crypto Engine

If the Cisco IOS crypto engine is encrypting traffic, but you want to cause an installed ESA crypto engine to encrypt the traffic instead, you can switch to the ESA crypto engine.



**Caution** Before you switch to the ESA crypto engine, be sure that the ESA crypto engine is configured with DSS keys generated and exchanged, and be sure that the DSS keys have the same *key-name* for both engines; otherwise, you will lose encryption capability when you switch engines.

To select the ESA crypto engine, perform the following task in global configuration mode:

Task	Command
Enable the ESA.	<b>crypto esa enable slot</b>

After you select the ESA crypto engine, the ESA crypto engine will be the active engine, governing the router interfaces. The ESA crypto engine will perform encryption services for the router, and the Cisco IOS crypto engine will be inactive.

### Delete DSS Keys

This is an optional task.

If you ever remove or relocate the ESA or the Cisco 7200, or if the DSS keys ever become compromised, or if you want to turn encryption off at the router, you might want to delete DSS keys to reduce any potential security risk. This section describes how to delete a DSS key pair for an ESA or for a Cisco 7200 series router.

To delete DSS keys, perform the following tasks beginning in EXEC mode:

Task	Command
View all existing sets of DSS keys (ESA and Cisco IOS keys).	<b>show crypto mypubkey</b>
Determine the current (active) crypto engine.	<b>show crypto engine configuration</b>
<b>If the current engine is not the engine for which you want to delete keys, change engines.</b> (When you delete keys, the software deletes keys for the current active engine.)	<b>crypto esa enable slot</b> (switch to the Cisco IOS crypto engine) or <b>crypto esa shutdown slot</b> (switch to the ESA crypto engine)
Verify that the current crypto engine is the engine for which you want to delete keys.	<b>show crypto engine configuration</b>
Delete the DSS keys for the current crypto engine.	<b>crypto zeroize</b> (for the Cisco IOS crypto engine) or <b>crypto zeroize slot</b> (for the ESA crypto engine)

After you delete DSS keys for a crypto engine, if you ever want to use that engine for encryption, you must regenerate and exchange new DSS keys for that engine. For the ESA crypto engine, you must also enable the ESA.

## Customize Encryption (Configure Options)

This section describes options that you can configure to customize encryption on a router:

- Define Time Duration of Encrypted Sessions
- Shorten Session Setup Times by Pregenerating DH Numbers
- Change Encryption Access List Limits

### Define Time Duration of Encrypted Sessions

The default time duration of an encrypted session is 30 minutes. After the default time duration expires, an encrypted session must be renegotiated if encrypted communication is to continue. You can change this default to extend or shorten the time of encrypted sessions.

You might want to shorten session times if you believe that there is a risk of compromised session keys.

You might want to extend session times if your system has trouble tolerating the interruptions caused when sessions are renegotiated.

To change the time duration of encrypted sessions, perform at least the first of the following tasks, in global configuration mode:

Task	Command
Define maximum time duration of encrypted sessions.	<b>crypto key-timeout</b> <i>minutes</i>
View defined time duration of encrypted sessions.	<b>show crypto key-timeout</b>

### Shorten Session Setup Times by Pregenerating DH Numbers

Diffie-Hellman (DH) numbers are generated in pairs during the setup of each encrypted session. (DH numbers are used during encrypted session setup to compute the DES session key.) Generating these numbers is a CPU-intensive activity, which can make session setup slow—especially for low-end routers. To speed up session setup time, you may choose to pregenerate DH numbers. It is usually necessary to pregenerate only one or two DH numbers.

To pregenerate DH numbers, perform the following task in global configuration mode:

Task	Command
Pregenerate DH numbers.	<b>crypto pregen-dh-pairs</b> <i>count</i> [ <i>slot</i> ]

### Change Encryption Access List Limits

When you configure encryption access lists, you configure source and destination pairs in criteria statements. Any traffic that matches the criteria is then encrypted.

By default, the maximum number of distinct sources (host or subnets) that you can define in an encryption access list is 100. Also, the maximum number of distinct destinations that you can define for any given source address is 10. For example, if you define six different source addresses, you can define up to 10 destination addresses for each of the six sources, for a total of 60 access list criteria statements.

### Why Do These Limits Exist?

These limits exist because of the amount of memory that must be reserved for encryption connections. If there are more potential connections, there must be more memory preallocated.

### When Should the Limits Be Changed?

For most situations, the defaults of 100 maximum sources and 10 maximum destinations per source are sufficient. Cisco recommends that you do not change the defaults unless you actually exceed the number of sources or destinations per source.

However, in some situations you might want to change one or both of these maximum values. For example, if more than 10 remote sites need to connect to one server behind your router, then you need more than 10 destination addresses (one for each remote site) to pair up with the server's source address in the local router's encryption access list. In this case, you need to change the default of 10 maximum destination addresses per source address.

When changing limits, you should consider the amount of memory that will be allocated. In general, if you increase one value, decrease the other value. This prevents your router from running out of memory because too much memory was preallocated.

### How Much Memory Is Preallocated If the Limits Are Changed?

The amount of memory reserved for encrypted connections changes if you change the defaults.

For every additional source, the following additional bytes of memory will be allocated:

$$64 + (68 \times \text{the specified number of maximum destinations})$$

For every additional destination, the following additional bytes of memory will be allocated:

$$68 \times \text{the specified number of maximum sources}$$

For example, if you specify 5 maximum sources, and 250 maximum destinations per source, the memory allocated for encryption connections is calculated as follows:

$$\{5 \times [64 + (68 \times 250)]\} + \{250 \times (68 \times 5)\} = 170320 \text{ bytes}$$

## How Are the Limits Changed?

To change the default limits, perform one or both of the following tasks in global configuration mode, then reboot the router for the changes to take effect:

Task	Command
Change the maximum number of distinct sources (hosts or subnets) that you can define in the encryption access list statements.	<b>crypto sdu entities</b> <i>number</i>
Change the maximum number of destinations (hosts or subnets) per source that you can define in the encryption access list statements.	<b>crypto sdu connections</b> <i>number</i>

**Note** You must reboot the router for these changes to take effect.

For an example of changing these values, see the section “Example of Changing Encryption Access List Limits,” later in this chapter.

## Turn Off Encryption

You can turn off encryption for certain router interfaces, or you can turn off encryption completely for the entire router.

- To turn off encryption at all the interfaces governed by a single crypto engine, you can delete DSS keys for that engine. Deleting DSS keys is described in this section.
- To turn off encryption at certain random interfaces, you can remove the crypto maps from the interfaces with the **no crypto map (interface configuration)** command.
- To turn off encryption completely for a router, you can delete the DSS keys for all the router’s crypto engines. Deleting DSS keys is described in this section.

Deleting DSS keys deconfigures encryption for the crypto engine and also reduces security risk by ensuring that the keys cannot be misused if you lose physical control over the router or ESA.

After you delete DSS keys for a crypto engine, you will not be able to perform encryption on the interfaces governed by that crypto engine.



**Caution** DSS keys cannot be recovered after they have been deleted. Use this function only after careful consideration.

For all platforms other than Cisco 7200 series routers, to delete DSS public/private keys for a crypto engine, perform the following task in global configuration mode:

Task	Command
Delete DSS keys for a crypto engine.	<b>crypto zeroize</b> [ <i>slot</i> ] <sup>1</sup>

1. Only Cisco 7200 and 7500 series routers require the *slot* argument.

For a Cisco 7200 series router, to delete DSS public/private keys for a crypto engine, refer to the section “Delete DSS Keys” earlier in this chapter.

## Testing and Troubleshooting Encryption

This section discusses how to verify your configuration and the correct operation of encryption. This section also discusses diagnosing encryption problems.

You should complete all the required configuration tasks (as described earlier in this chapter) before trying to test or troubleshoot your encryption configuration.

This section includes the following topics:

- Test the Encryption Configuration
- Diagnose Connection Problems
- Diagnose Other Miscellaneous Problems
- Use Debug Commands

### Test the Encryption Configuration

If you want to test the encryption setup between peer routers, you can attempt to manually establish a session using the IP address of a local host and a remote host which have been specified in an encryption access list. (The encryption list must be specified in a crypto map definition, and that crypto map must be applied to an interface before this test will be successful.)

To test the encryption setup, perform the following tasks in privileged EXEC mode:

Task	Command
Set up a test encryption session.	<b>test crypto initiate-session</b> <i>src-ip-addr dst-ip-addr map-name seq-num</i>
View the connection status.	<b>show crypto connections</b>

An example at the end of this chapter explains how to interpret the **show crypto connections** command output.

### Diagnose Connection Problems

If you need to verify the state of a connection, you can perform the following tasks in privileged EXEC mode:

Task	Command
Check status of all encryption connections.	<b>show crypto connections</b>
Check status of a crypto map.	<b>show crypto map</b>
Check that connection is established and that packets are being encrypted.	<b>show crypto engine connections active</b>

### Diagnose Other Miscellaneous Problems

When using encryption, you might encounter some of these problems:

- Dropped Packets
- Difficulty Establishing Telnet Sessions
- Invalid DSS Public/Private Keys
- ESA Crypto Engine Not Active

- Password Requested When You Generate DSS Keys
- Router Hanging

## Dropped Packets

Packets are normally dropped while an encrypted session is being set up. If this poses a problem for your network, you should extend the length of encryption sessions as described previously in the section “Define Time Duration of Encrypted Sessions.” The longer the session time, the fewer the interruptions caused by session renegotiation.

Packets might also be dropped if you switch crypto engines in a Cisco 7200 series router with an ESA. If this is a problem, you should only switch crypto engines when encrypted traffic is light.

IP fragments are always dropped on VIP2 interfaces, because IP fragmentation is not supported with encryption on VIP2 interfaces.

## Difficulty Establishing Telnet Sessions

Hosts might experience difficulty in establishing Telnet sessions if the session uses two encrypting peer routers to create the connection. This difficulty is more likely to occur if the peer routers are low-end routers such as Cisco 2500 series routers. Telnet sessions can fail to be established when a Telnet connection attempt times out before the encrypted session setup is complete.

If a Telnet session fails to establish, the host should wait a short time (a few seconds might be sufficient), and then attempt the Telnet connection again. After the short wait, the encrypted session setup should be complete, and the Telnet session can be established. Enabling pregeneration of DH numbers (described later in this chapter) might also help by speeding up encryption session connection setup times.

## Invalid DSS Public/Private Keys

If NVRAM fails, or if your ESA is tampered with or replaced, DSS public/private keys will no longer be valid. If this happens, you will need to regenerate and re-exchange DSS keys. Generating and exchanging DSS keys are described earlier in the section “Configure Encryption.”

## ESA Crypto Engine Not Active

If an installed ESA is not active when you boot a router, the router displays a message similar to this message, indicating that the router switched over to the Cisco IOS crypto engine:

```
There are no keys on the ESA in slot 2- ESA not enabled

...switching to SW crypto engine
```

You can also determine if the ESA crypto engine is not active by using the **show crypto engine brief** command—look at the “crypto engine state” field in the output. If no crypto engine is active, the state field indicates “pending.”

The ESA crypto engine will not be active if you removed and reinstalled the ESA, if the ESA was tampered with, or if encryption is not configured correctly for the ESA.

If the Cisco IOS crypto engine is active, but you want to use the ESA crypto engine instead, make sure that the ESA crypto engine is reset (**crypto clear-latch** command), and for Cisco 7200 series routers, also make sure that the ESA crypto engine is enabled (**crypto esa enable** command). You

might also need to complete or verify additional configuration; refer to the instructions for configuring encryption with an ESA, in the earlier section “Configure Encryption with an ESA in a VIP2” or “Configure Encryption with an ESA in a Cisco 7200 Series Router.”

To verify that the ESA has DSS keys, you can use the **show crypto card** command and look at the “DSS Key set” field in the output. If the field contains “Yes,” the ESA has DSS keys generated and stored. In this case, you might only need to reset and enable the ESA to make it active.

### Password Requested When You Generate DSS Keys

- If you attempt to generate DSS keys for the Cisco IOS crypto engine on a Cisco 7200 series router with an installed ESA without DSS keys, the router might assume that you are trying to generate keys for the ESA and prompt for the ESA password.
  - If you want to generate keys for the ESA, you must supply the ESA password. If you do not know the password, you must reset the ESA as described in the section “Configure Encryption with an ESA in a Cisco 7200 Series Router” (earlier in this chapter).
  - If you want to generate keys for the Cisco IOS crypto engine, not the ESA crypto engine, you must select the Cisco IOS crypto engine to make it the active engine.

To select the Cisco IOS crypto engine, perform the following task in global configuration mode:

Task	Command
Shut down the ESA.	<b>crypto esa shutdown slot</b>

- When the Cisco IOS crypto engine is active, you can generate keys for the router, and you will not be prompted for a password.
- If you want to generate DSS keys for an ESA in a VIP2 (for Cisco RSP7000 and 7500 series routers), you must enter the ESA password. If you do not know the password, you must reset the ESA as described in the section “Configure Encryption with an ESA in a VIP2” (earlier in this chapter).

### Router Hanging

If you remove a configured ESA from a VIP2, you must reboot the router. If you do not, the router might hang when it tries to access the absent ESA.

### Use Debug Commands

Debug commands are also available to assist in problem solving. These commands are documented in the *Debug Command Reference*.

## Encryption Configuration Examples

The following sections provide examples of configuring and testing your router for network data encryption:

- Example of Generating DSS Public/Private Keys
- Example of Exchanging DSS Public Keys
- Example of Enabling DES Encryption Algorithms

- Examples of Setting Up Encryption Access Lists, Defining Crypto Maps, and Applying Crypto Maps to Interfaces
- Example of Changing Encryption Access List Limits
- Examples of Configuring Encryption With GRE Tunnels
- Examples of ESA-Specific Encryption Configuration Tasks
- Examples of Deleting DSS Keys
- Example of Testing the Encryption Connection

## Example of Generating DSS Public/Private Keys

The following example illustrates two encrypting peer routers (named Apricot and Banana) generating their respective DSS public/private keys. Apricot is a Cisco 2500 series router. Banana is a Cisco 7500 series router with an RSP in chassis slot 4 and an ESA/VIP2 in chassis slot 2.

### Apricot

```
Apricot(config)# crypto gen-signature-keys Apricot
Generating DSS keys .... [OK]
Apricot(config)#
```

### Banana

```
Banana(config)# crypto gen-signature-keys BananaIOS 4
Generating DSS keys .... [OK]
Banana(config)# crypto gen-signature-keys BananaESA 2
% Initialize the crypto card password. You will need
  this password in order to generate new signature
  keys or clear the crypto card extraction latch.

Password: <passwd>

Re-enter password: <passwd>

Generating DSS keys .... [OK]
Banana(config)#
```

The password entered in this example is a new password that you create when you generate DSS keys for an ESA crypto engine for the first time. If you ever generate DSS keys a second time for the same ESA crypto engine, you must use the same password to complete the key regeneration.

## Example of Exchanging DSS Public Keys

The following is an example of a DSS public key exchange between two peer encrypting routers (Apricot and Banana). Apricot is a Cisco 2500 series router, and Banana is a Cisco 7500 series router with an ESA. In this example, Apricot sends its Cisco IOS DSS public key, and Banana sends its ESA DSS public key. DSS keys have already been generated as shown in the previous example.

Before any commands are entered, one administrator must call the other administrator. After the phone call is established, the two administrators decide which router is “PASSIVE” and which is “ACTIVE” (an arbitrary choice). In this example, router Apricot is ACTIVE and router Banana is PASSIVE. To start, PASSIVE enables a connection as follows:

### Banana (PASSIVE)

```
Banana(config)# crypto key-exchange passive
Enter escape character to abort if connection does not complete.
Wait for connection from peer[confirm]<Return>
Waiting ....
```

PASSIVE must wait while ACTIVE initiates the connection and sends a DSS public key.

### Apricot (ACTIVE)

```
Apricot(config)# crypto key-exchange 192.168.114.68 Apricot
Public key for Apricot:
  Serial Number 01461300
  Fingerprint   0F1D 373F 2FC1 872C D5D7

Wait for peer to send a key[confirm]<Return>
Waiting ....
```

After ACTIVE sends a DSS public key, the key’s serial number and Fingerprint display on both terminals, as shown previously and as follows:

### Banana (PASSIVE)

```
Public key for Apricot:
  Serial Number 01461300
  Fingerprint   0F1D 373F 2FC1 872C D5D7
Add this public key to the configuration? [yes/no]: y
```

Now the two administrators both must verbally verify that their two screens show the same serial number and Fingerprint. If they do, PASSIVE will accept the DSS key as shown previously by typing **y**, and continue by sending ACTIVE a DSS public key:

```
Send peer a key in return[confirm]<Return>
Which one?

BananaIOS? [yes]: n
BananaESA? [yes]: <Return>
Public key for BananaESA:
  Serial Number 01579312
  Fingerprint   BF1F 9EAC B17E F2A1 BA77

Banana(config)#
```

Both administrators observe Banana’s serial number and Fingerprint on their screens. Again, they verbally verify that the two screens show the same numbers.

**Apricot (ACTIVE):**

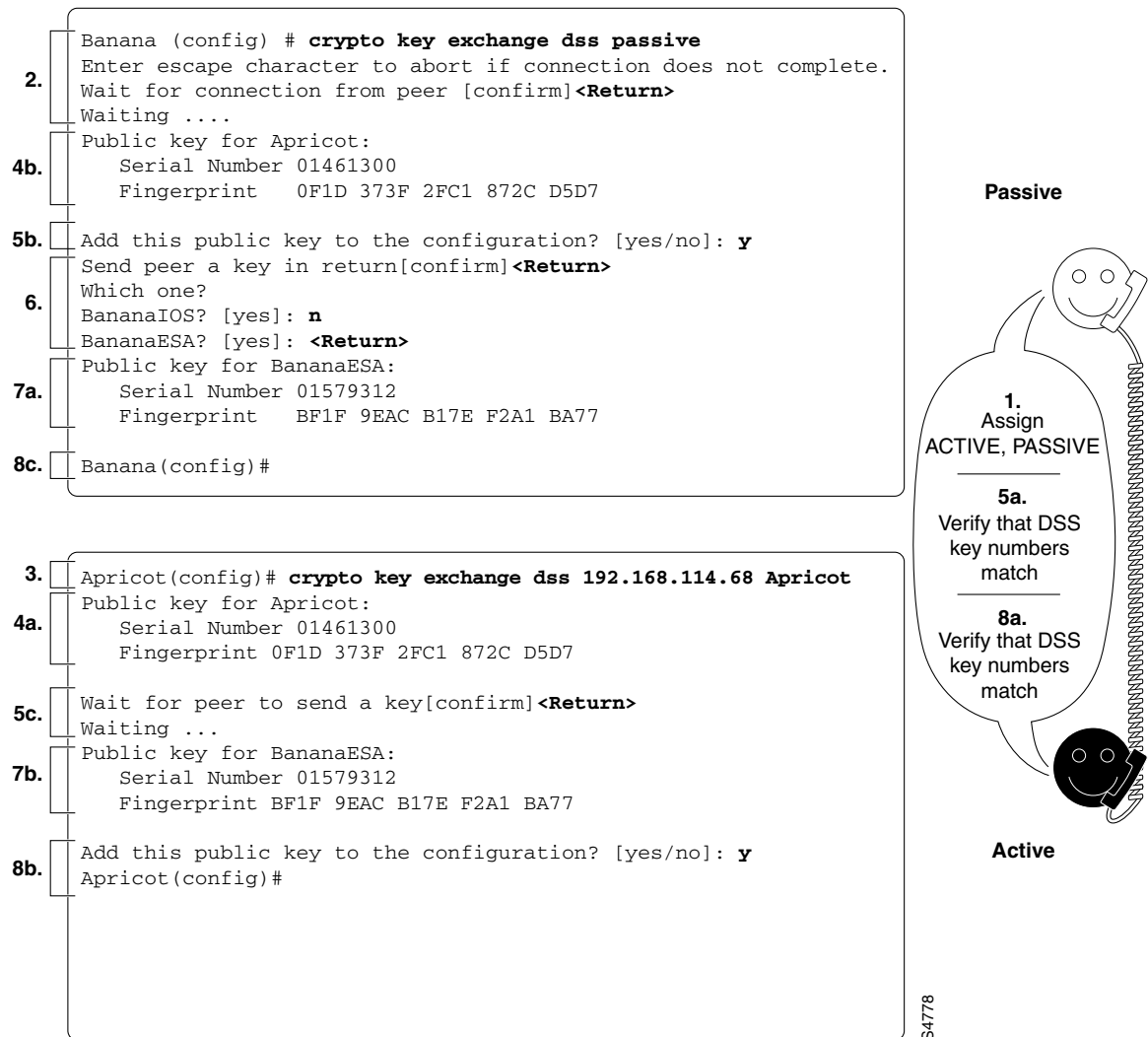
```
Public key for BananaESA:
  Serial Number 01579312
  Fingerprint   BF1F 9EAC B17E F2A1 BA77
```

```
Add this public key to the configuration? [yes/no]: y
Apricot(config)#
```

ACTIVE accepts Apricot's DSS public key. Both administrators hang up the phone and the key exchange is complete.

Figure 14 shows complete screens of the two routers. The steps are numbered on the figure to show the sequence of the entire exchange.

**Figure 14 DSS Public Key Exchange (Numbers Indicate Sequence of Events)**



## Example of Enabling DES Encryption Algorithms

In this example, a router (Apricot) globally enables two DES algorithms: the basic DES algorithm with 8-bit Cipher Feedback (CFB), and the 40-bit DES algorithm with 8-bit CFB. Another router (Banana) globally enables three DES algorithms: the basic DES algorithm with 8-bit CFB, the basic DES algorithm with 64-bit CFB, and the 40-bit DES algorithm with 8-bit CFB.

The following commands are entered from the global configuration mode.

### Apricot

```
crypto algorithm des cfb-8  
crypto algorithm 40-bit-des cfb-8
```

### Banana

```
crypto algorithm des cfb-8  
crypto algorithm des cfb-64  
crypto algorithm 40-bit-des cfb-8
```

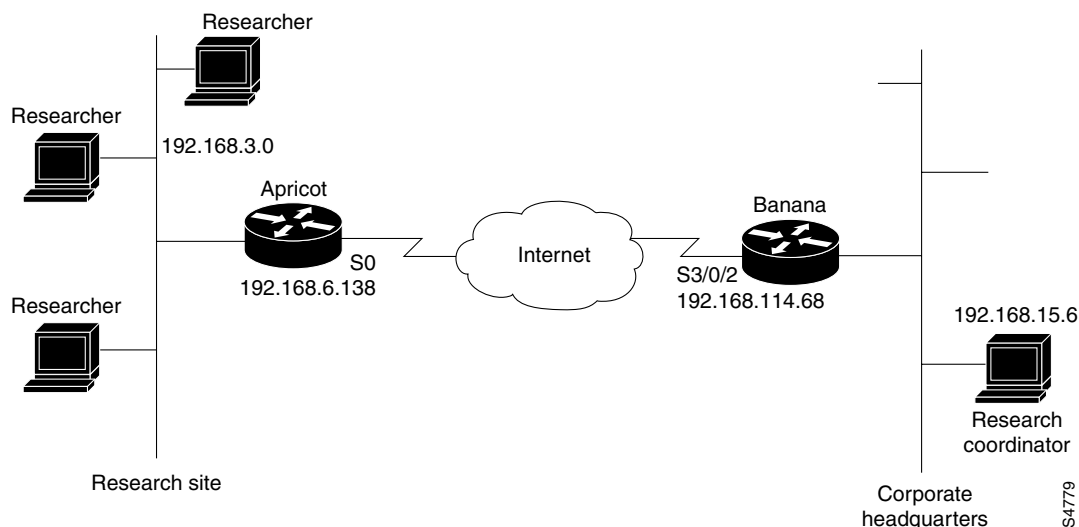
## Examples of Setting Up Encryption Access Lists, Defining Crypto Maps, and Applying Crypto Maps to Interfaces

The following two examples show how to set up interfaces for encrypted transmission. Participating routers will be configured as encrypting peers for IP packet encryption.

### Example 1

In the first example, a team of researchers at a remote site communicate with a research coordinator at headquarters. Company-confidential information is exchanged by IP traffic that consists only of TCP data. Figure 15 shows the network topology.

Figure 15 Example 1 Network Topology



Apricot is a Cisco 2500 series router, and Banana is a Cisco 7500 series router with an ESA/VIP2 in chassis slot 3.

### Apricot

```
Apricot(config)# access-list 101 permit tcp 192.168.3.0 0.0.0.15 host 192.168.15.6
Apricot(config)# crypto map Research 10
Apricot(config-crypto-map)# set peer BananaESA
Apricot(config-crypto-map)# set algorithm des cfb-8
Apricot(config-crypto-map)# match address 101
Apricot(config-crypto-map)# exit
Apricot(config)# interface s0
Apricot(config-if)# crypto map Research
Apricot(config-if)# exit
Apricot(config)#
```

### Banana

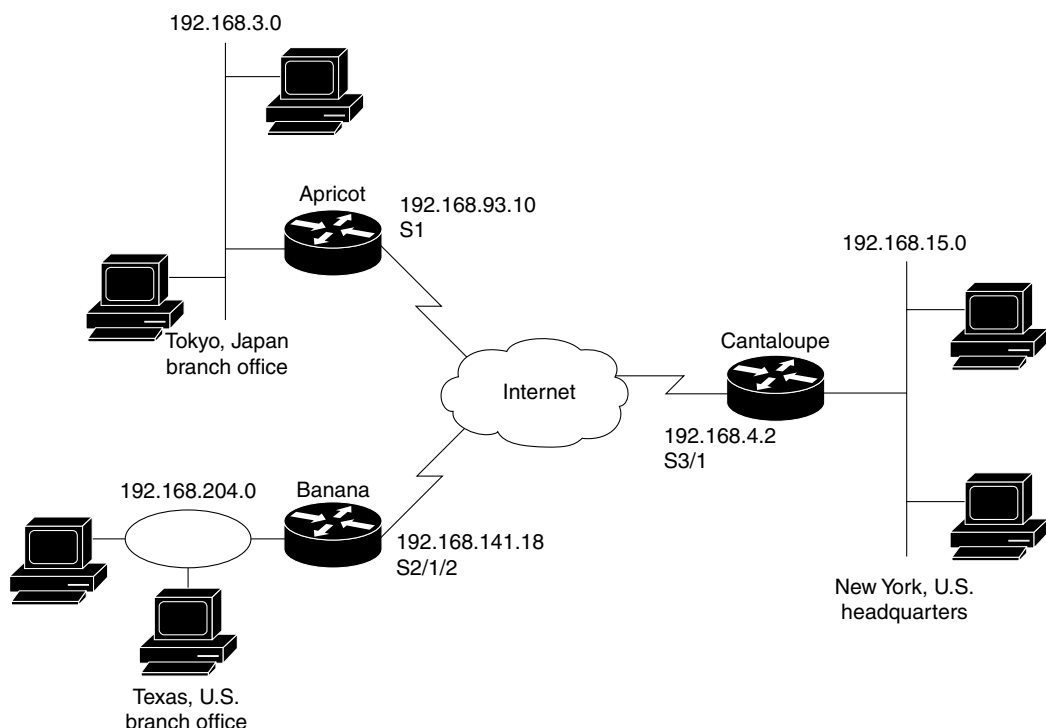
```
Banana(config)# access-list 110 permit tcp host 192.168.15.6 192.168.3.0 0.0.0.15
Banana(config)# crypto map Rsrch 10
Banana(config-crypto-map)# set peer Apricot
Banana(config-crypto-map)# set algorithm des cfb-8
Banana(config-crypto-map)# set algorithm des cfb-64
Banana(config-crypto-map)# match address 110
Banana(config-crypto-map)# exit
Banana(config)# interface s3/0/2
Banana(config-if)# crypto map Rsrch
Banana(config-if)# exit
Banana(config)#
```

Because Banana set two DES algorithms for crypto map Rsrch, Banana could use either algorithm with traffic on the S3/0/2 interface. However, because Apricot only set one DES algorithm (CFB-8 DES) for the crypto map Research, that is the only DES algorithm that will be used for all encrypted traffic between Apricot and Banana.

## Example 2

In the second example, employees at two branch offices and at headquarters must communicate sensitive information. A mix of TCP and UDP traffic is transmitted by IP packets. Figure 16 shows the network topology used in this example.

Figure 16 Example 2 Network Topology



S4780

Apricot is a Cisco 2500 series router and connects to the Internet through interface S1. Both Banana and Cantaloupe are Cisco 7500 series routers with ESA cards. Banana connects to the Internet using the ESA-governed VIP2 interface S2/1/2. Cantaloupe is already using every VIP2 interface (governed by the ESA card) to connect to several offsite financial services, so it must connect to the Internet using a serial interface (S3/1) in slot 3. (Cantaloupe’s interface S3/1 is governed by the Cisco IOS crypto engine.)

Apricot will be using one interface to communicate with both Banana and Cantaloupe. Because only one crypto map can be applied to this interface, Apricot creates a crypto map that has two distinct definition sets by using two different *seq-num* values with the same *map-name*. By using *seq-num* values of 10 and 20, Apricot creates a single crypto map named “TXandNY” that contains a subset of definitions for encrypted sessions with Banana, and a second distinct subset for definitions for encrypted sessions with Cantaloupe.

Banana and Cantaloupe each also use a single interface to communicate with the other two routers, and therefore will use the same strategy as Apricot does for creating crypto maps.

In this example, we assume that Apricot has generated DSS keys with the *key-name* “Apricot.TokyoBranch,” Banana has generated DSS keys with the *key-name* “BananaESA.TXbranch,” and Cantaloupe has generated DSS keys with the *key-name* “CantaloupeIOS.NY.” We also assume that each router has exchanged DSS public keys with the other two routers, and that each router has enabled each DES algorithm that is specified in the crypto maps.

## Apricot

```
Apricot(config)# access-list 105 permit tcp 192.168.3.0 0.0.0.15 192.168.204.0 0.0.0.255
Apricot(config)# access-list 105 permit udp 192.168.3.0 0.0.0.15 192.168.204.0 0.0.0.255
Apricot(config)# access-list 106 permit tcp 192.168.3.0 0.0.0.15 192.168.15.0 0.0.0.255
Apricot(config)# access-list 106 permit udp 192.168.3.0 0.0.0.15 192.168.15.0 0.0.0.255
Apricot(config)# crypto map TXandNY 10
Apricot(config-crypto-map)# set peer BananaESA.TXbranch
Apricot(config-crypto-map)# set algorithm 40-bit-des cfb-8
Apricot(config-crypto-map)# match address 105
Apricot(config-crypto-map)# exit
Apricot(config)# crypto map TXandNY 20
Apricot(config-crypto-map)# set peer CantaloupeIOS.NY
Apricot(config-crypto-map)# set algorithm 40-bit-des cfb-64
Apricot(config-crypto-map)# match address 106
Apricot(config-crypto-map)# exit
Apricot(config)# interface s1
Apricot(config-if)# crypto map TXandNY
Apricot(config-if)# exit
Apricot(config)#
```

## Banana

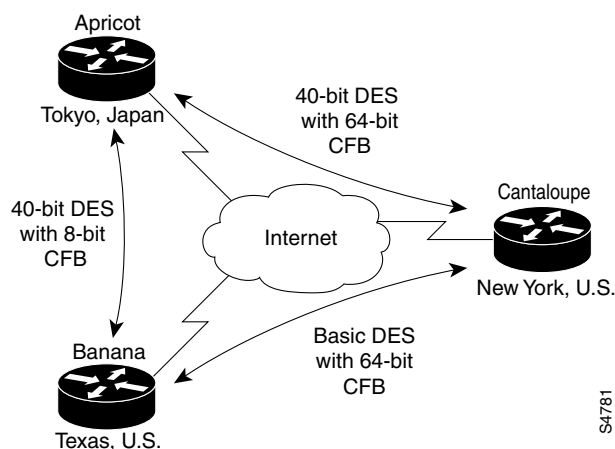
```
Banana(config)# access-list 110 permit tcp 192.168.204.0 0.0.0.255 192.168.3.0 0.0.0.15
Banana(config)# access-list 110 permit udp 192.168.204.0 0.0.0.255 192.168.3.0 0.0.0.15
Banana(config)# access-list 120 permit tcp 192.168.204.0 0.0.0.255 192.168.15.0 0.0.0.255
Banana(config)# access-list 120 permit udp 192.168.204.0 0.0.0.255 192.168.15.0 0.0.0.255
Banana(config)# crypto map USA 10
Banana(config-crypto-map)# set peer Apricot.TokyoBranch
Banana(config-crypto-map)# set algorithm 40-bit-des cfb-8
Banana(config-crypto-map)# match address 110
Banana(config-crypto-map)# exit
Banana(config)# crypto map USA 20
Banana(config-crypto-map)# set peer CantaloupeIOS.NY
Banana(config-crypto-map)# set algorithm des cfb-64
Banana(config-crypto-map)# match address 120
Banana(config-crypto-map)# exit
Banana(config)# interface s2/1/2
Banana(config-if)# crypto map USA
Banana(config-if)# exit
Banana(config)#
```

## Cantaloupe

```
Cantaloupe(config)# access-list 101 permit tcp 192.168.15.0 0.0.0.255 192.168.3.0 0.0.0.15
Cantaloupe(config)# access-list 101 permit udp 192.168.15.0 0.0.0.255 192.168.3.0 0.0.0.15
Cantaloupe(config)# access-list 102 permit tcp 192.168.15.0 0.0.0.255 192.168.204.0 0.0.0.255
Cantaloupe(config)# access-list 102 permit udp 192.168.15.0 0.0.0.255 192.168.204.0 0.0.0.255
Cantaloupe(config)# crypto map satellites 10
Cantaloupe(config-crypto-map)# set peer Apricot.TokyoBranch
Cantaloupe(config-crypto-map)# set algorithm 40-bit-des cfb-64
Cantaloupe(config-crypto-map)# match address 101
Cantaloupe(config-crypto-map)# exit
Cantaloupe(config)# crypto map satellites 20
Cantaloupe(config-crypto-map)# set peer BananaESA.TXbranch
Cantaloupe(config-crypto-map)# set algorithm des cfb-64
Cantaloupe(config-crypto-map)# match address 102
Cantaloupe(config-crypto-map)# exit
Cantaloupe(config)# interface s3/1
Cantaloupe(config-if)# crypto map satellites
Cantaloupe(config-if)# exit
Cantaloupe(config)#
```

The previous configurations will result in DES encryption algorithms being applied to encrypted IP traffic as shown in Figure 17.

**Figure 17 Example 2 DES Encryption Algorithms**



### Example of Changing Encryption Access List Limits

In this example, there are 50 remote sites connecting to a single server. The connections between the server and each site need to be encrypted. The server is located behind the local router named Apricot. Each of the remote sites connects through its own router.

Because of the large number of destination addresses that must be paired with the same source address in the local encryption access list, the default limits are changed.

```
Apricot(config)# crypto sdu connections 60
%Please reboot for the new connection size to take effect

Apricot(config)# crypto sdu entities 5
%Please reboot for the new table size to take effect
```

Even though there is only one server, and only 50 remote sites, this example defines 5 sources and 60 destinations. This allows room for future growth of the encryption access list. If another source or destination is added later, the limits will not have to be increased and the router rebooted again, which is a disruptive process.

### Examples of Configuring Encryption With GRE Tunnels

There are two example configurations for encryption with GRE tunnels:

- Example of Encrypting Only GRE Tunnel Traffic
- Example of Encrypting Both GRE Tunnel Traffic and Other Non-GRE Traffic

## Example of Encrypting Only GRE Tunnel Traffic

This configuration causes all traffic through the GRE tunnel to be encrypted. No other traffic at the interface will be encrypted. The GRE tunnel is from router Apricot to router Banana. (Only partial configuration files are shown for each router.)

### Apricot

```
crypto map BananaMap 10
  set algorithm 40-bit-des
  set peer Banana
  match address 101
!
interface Tunnel0
  no ip address
  ipx network 923FA800
  tunnel source 10.1.1.2
  tunnel destination 10.1.1.1
  crypto map BananaMap
!
interface Serial0
  ip address 10.1.1.2 255.255.255.0
  crypto map BananaMap
!
access-list 101 permit gre host 10.1.1.2 host 10.1.1.1
```

### Banana

```
crypto map ApricotMap 10
  set algorithm 40-bit-des
  set peer Apricot
  match address 102
!
interface Tunnel0
  no ip address
  ipx network 923FA800
  tunnel source 10.1.1.1
  tunnel destination 10.1.1.2
  crypto map ApricotMap
!
interface Serial0
  ip address 10.1.1.1 255.255.255.0
  clockrate 2000000
  no cdp enable
  crypto map ApricotMap
!
access-list 102 permit gre host 10.1.1.1 host 10.1.1.2
```

### Example of Encrypting Both GRE Tunnel Traffic and Other Non-GRE Traffic

This configuration encrypts all GRE tunnel traffic, and it also encrypts TCP traffic between two hosts with the IP addresses 172.16.25.3 and 192.168.3.5. The GRE tunnel is from router Apricot to router Banana. (Only partial configuration files are shown for each router.)

#### Apricot

```
crypto map BananaMapTunnel 10
  set algorithm 40-bit-des
  set peer Banana
  match address 101
!
crypto map BananaMapSerial 10
  set algorithm 40-bit-des
  set peer Banana
  match address 101
crypto map BananaMapSerial 20
  set algorithm 40-bit-des
  set peer Banana
  match address 110
!
interface Tunnel0
  no ip address
  ipx network 923FA800
  tunnel source 10.1.1.2
  tunnel destination 10.1.1.1
  crypto map BananaMapTunnel
!
interface Serial0
  ip address 10.1.1.2 255.255.255.0
  crypto map BananaMapSerial
!
access-list 101 permit gre host 10.1.1.2 host 10.1.1.1
access-list 110 permit tcp host 172.16.25.3 host 192.168.3.5
```

#### Banana

```
crypto map ApricotMapTunnel 10
  set algorithm 40-bit-des
  set peer Apricot
  match address 102
!
crypto map ApricotMapSerial 10
  set algorithm 40-bit-des
  set peer Apricot
  match address 102
crypto map ApricotMapSerial 20
  set algorithm 40-bit-des
  set peer Apricot
  match address 112
!
interface Tunnel0
  no ip address
  ipx network 923FA800
  tunnel source 10.1.1.1
  tunnel destination 10.1.1.2
  crypto map ApricotMapTunnel
!
```

```

interface Serial0
 ip address 10.1.1.1 255.255.255.0
 clockrate 2000000
 no cdp enable
 crypto map ApricotMapSerial
 !
 access-list 102 permit gre host 10.1.1.1 host 10.1.1.2
 access-list 112 permit tcp host 192.168.3.5 host 172.16.25.3

```

## Examples of ESA-Specific Encryption Configuration Tasks

This section includes examples of the following:

- Examples of Resetting an ESA
- Example of Enabling an ESA (Cisco 7200 Series Routers Only)
- Example of Selecting a Different Crypto Engine (Cisco 7200 Series Routers Only)

### Examples of Resetting an ESA

The following example resets an ESA on a Cisco 7500 series router. The ESA is in a VIP2 that is in slot 4 of the router chassis.

```

Banana(config)# crypto clear-latch 4
% Enter the crypto card password.
Password: <passwd>
Banana(config)#

```

The following example resets an ESA without DSS keys, for a Cisco 7200 series router. The ESA is in the router chassis slot 2.

```

Apricot(config)# crypto clear-latch 2
% Enter the crypto card password.
Password: <passwd>
ESA in slot 2 not enabled.
[OK]
Apricot(config)#

```

The following example resets an ESA with DSS keys, for a Cisco 7200 series router; the ESA was previously in use on the same router, but was removed and reinstalled with OIR. No changes to the encryption configuration are desired by the administrator. The ESA is in the router chassis slot 2.

```

Apricot(config)# crypto clear-latch 2
% Enter the crypto card password.
Password: <passwd>
Keys were found for this ESA- enable ESA now? [yes/no]: yes
...switching to HW crypto engine
[OK]
Apricot(config)#

```

The following example resets an ESA with DSS keys, for a Cisco 7200 series router; the ESA was previously used in a different router, and requires new DSS keys to be generated and exchanged before the ESA can become operational. The ESA is in the router chassis slot 2.

```

Apricot(config)# crypto clear-latch 2
% Enter the crypto card password.
Password: <passwd>
Keys were found for this ESA- enable ESA now? [yes/no]: no
ESA in slot 2 not enabled.
[OK]
Apricot(config)#

```

### Example of Enabling an ESA (Cisco 7200 Series Routers Only)

The following example enables an ESA in the router chassis slot 2:

```
Apricot(config)# crypto esa enable 2  
...switching to HW crypto engine  
Apricot(config)#
```

### Example of Selecting a Different Crypto Engine (Cisco 7200 Series Routers Only)

Select a different crypto engine only if the new engine is fully configured for encryption.

The following example switches from the Cisco IOS crypto engine to the ESA crypto engine. The ESA crypto engine is in the router chassis slot 4.

```
Apricot(config)# crypto esa enable 4  
...switching to HW crypto engine  
Apricot(config)#
```

The following example switches from the ESA crypto engine to the Cisco IOS crypto engine. The ESA crypto engine is in the router chassis slot 4.

```
Apricot(config)# crypto esa shutdown 4  
...switching to SW crypto engine  
Apricot(config)#
```

## Examples of Deleting DSS Keys

This section includes an example for a Cisco 7500 series router and an example for a Cisco 7200 series router with an installed ESA.

### Example for a Cisco 7500 Series Router

The following example deletes all the DSS keys on a Cisco 7500 series router. The RSP is in chassis slot 3 and a VIP2 is in chassis slot 4. Deleting all the DSS keys turns off encryption completely for the router. The Cisco IOS crypto engine keys are deleted first, then the VIP2 crypto engine keys.

```
Apricot(config)# crypto zeroize 3  
Warning! Zeroize will remove your DSS signature keys.  
Do you want to continue? [yes/no]: y  
Keys to be removed are named Apricot.IOS.  
Do you really want to remove these keys? [yes/no]: y  
[OK]  
Apricot(config)# crypto zeroize 4  
Warning! Zeroize will remove your DSS signature keys.  
Do you want to continue? [yes/no]: y  
Keys to be removed are named Apricot.VIP.  
Do you really want to remove these keys? [yes/no]: y  
[OK]  
Apricot(config)#
```

## Example for a Cisco 7200 Series Router

The following example deletes DSS keys only for an ESA, in chassis slot 2 of a Cisco 7200 series router. The Cisco IOS crypto engine DSS keys are not deleted in this example.

### 1 View existing DSS keys:

```
Apricot# show crypto mypubkey
crypto public-key Apricot.IOS 01709642
BDD99A6E EEE53D30 BC0BFAE6 948C40FB 713510CB 32104137 91B06C8D C2D5B422
D9C154CA 00CDE99B 425DB9FD FE3162F1 1E5866AF CF66DD33 677259FF E5C24812
quit
crypto public-key Apricot.ESA 01234567
866AFCF6 E99B425D FDFE3162 BC0BFAE6 13791B06 713510CB 4CA00CDE 0BC0BFAE
3791B06C 154C0CDE F11E5866 AE6948C4 DD336772 3F66DF33 355459FF 2350912D
quit

Apricot#
```

This output shows that DSS keys exist for both the Cisco IOS crypto engine and for the ESA crypto engine.

### 2 Determine the Active Crypto Engine:

```
Apricot# show crypto engine configuration
engine name:      Apricot.IOS
engine type:      software
serial number:    01709642
platform:         rsp crypto engine

Encryption Process Info:
input queue top:  44
input queue bot:  44
input queue count: 0

Apricot#
```

The output shows that the Cisco IOS crypto engine is the active engine.

### 3 Because we want to delete DSS keys for the ESA crypto engine, change to the ESA crypto engine:

```
Apricot# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Apricot(config)# crypto esa enable 2
...switching to HW crypto engine
Apricot(config)#
```

### 4 Verify that the ESA crypto engine is the active engine:

```
Apricot(config)# exit
Apricot# show crypto engine configuration
engine name:      Apricot.ESA
engine type:      hardware
serial number:    01234567
platform:         esa crypto engine

Encryption Process Info:
input queue top:  0
input queue bot:  0
input queue count: 0

Apricot#
```

The output shows that the ESA crypto engine is now the active engine.

### 5 Delete the ESA DSS keys:

```
Apricot# config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Apricot(config)# crypto zeroize 2
Warning! Zeroize will remove your DSS signature keys.
Do you want to continue? [yes/no]: y
Keys to be removed are named Apricot.ESA.
Do you really want to remove these keys? [yes/no]: y
[OK]

Apricot(config)#
```

### 6 View existing DSS keys:

```
Apricot(config)# exit
Apricot# show crypto mypubkey
crypto public-key Apricot.IOS 01709642
BDD99A6E EEE53D30 BC0BFAE6 948C40FB 713510CB 32104137 91B06C8D C2D5B422
D9C154CA 00CDE99B 425DB9FD FE3162F1 1E5866AF CF66DD33 677259FF E5C24812
quit

Apricot#
```

The output shows that the ESA crypto engine keys have been deleted.

### 7 Determine the active crypto engine:

```
Apricot# show crypto engine configuration
engine name:      Apricot.IOS
engine type:      software
serial number:    01709642
platform:         rsp crypto engine

Encryption Process Info:
input queue top:  0
input queue bot:  0
input queue count: 0

Apricot#
```

The output shows that the system has defaulted back to the Cisco IOS crypto engine as the active engine.

## Example of Testing the Encryption Connection

The following example sets up and verifies a test encryption session.

Assume the same network topology and configuration as in the previous example and shown in Figure 16.

In this example, router Apricot sets up a test encryption session with router Banana and then views the connection status to verify a successful encrypted session connection.

#### Step 1 Router Apricot sets up a test encryption connection with router Banana.

```
Apricot# test crypto initiate-session 192.168.3.12 192.168.204.110 BananaESA.TXbranch 10
Sending CIM to: 192.168.204.110 from: 192.168.3.12.
Connection id: -1
```

Notice the Connection id value is -1. A negative value indicates that the connection is being set up.

**Step 2** Router Apricot issues the **show crypto connections** command.

```
Apricot# show crypto connections
Pending Connection Table
PE                UPE                Timestamp                Conn_id
192.168.3.10      192.168.204.100    Mar 01 1993 00:01:09     -1

Connection Table
PE                UPE                Conn_id New_id  Alg    Time
192.168.3.10      192.168.204.100    -1      1      0      Not Set
flags:PEND_CONN
```

Look in the Pending Connection Table for an entry with a Conn\_id value equal to the previously shown Connection id value—in this case, look for an entry with a Conn\_id value of -1. If this is the first time an encrypted connection has been attempted, there will only be one entry (as shown).

Note the PE and UPE addresses for this entry.

**Step 3** Now, look in the Connection Table for an entry with the same PE and UPE addresses. In this case, there is only one entry in both tables, so finding the right Connection Table entry is easy.

**Step 4** At the Connection Table entry, note the Conn\_id and New\_id values. In this case, Conn\_id equals -1 (as in the Pending Connection Table), and New\_id equals 1. The New\_id value of 1 will be assigned to the test connection when setup is complete. (Positive numbers are assigned to established, active connections.)

**Step 5** Apricot waits a few seconds for the test connection to establish and then reissues the **show crypto connections** command.

```
Apricot# show crypto connections
Connection Table
PE                UPE                Conn_id New_id  Alg    Time
192.168.3.10      192.168.204.100    1       0      0      Mar 01 1993 00:02:00
flags:TIME_KEYS
```

Again, look for the Connection Table entry with the same PE and UPE addresses as shown before. In this entry, notice that the Conn\_id value has changed to 1. This indicates that our test connection has been successfully established, because the Conn\_id value has changed to match the New\_id value of Step 4. Also, New\_id has been reset to 0 at this point, indicating that there are no new connections currently being set up.

In the command output of Step 5, there is no longer a Pending Connection Table being displayed, which indicates that there are currently no pending connections. This is also a good clue that the test connection was successfully established.

The **show crypto connections** command is explained in greater detail in the chapter “Network Data Encryption Commands” in the *Security Command Reference*. There you can find a description of how connection id’s are assigned during and following connection setup.

