Security for VPNs with IPsec Configuration Guide Cisco IOS XE Release 2

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Configuring Security for VPNs with IPsec

This module describes how to configure basic IP Security (IPsec) VPNs. IPsec is a framework of open standards developed by the Internet Engineering Task Force (IETF). It provides security for transmission of sensitive information over unprotected networks such as the Internet. IPsec acts at the network layer, protecting and authenticating IP packets between participating IPsec devices ("peers"), such as Cisco routers.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

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

Prerequisites for Configuring Security for VPNs with IPsec

IKE Configuration

You must configure Internet Key Exchange (IKE) as described in the module "Configuring Internet Key Exchange Security for IPsec VPNs."

Even if you decide to not use IKE, you must disable it as described in the module "Configuring Internet Key Exchange for IPsec VPNs."

Ensure Access Lists Are Compatible with IPsec

IKE uses UDP port 500. The IPsec Encapsulating Security Payload (ESP) and Authentication Header (AH) protocols use protocol numbers 50 and 51. Ensure that your access lists are configured so that protocol 50, 51, and User Datagram Protocol (UDP) port 500 traffic is not blocked at interfaces used by IPsec. In some cases, you might need to add a statement to your access lists to explicitly permit this traffic.

Restrictions for Configuring Security for VPNs with IPsec

Unicast IP Datagram Application Only

At this time, IPsec can be applied to unicast IP datagrams only. Because the IPsec Working Group has not yet addressed the issue of group key distribution, IPsec does not currently work with multicasts or broadcast IP datagrams.

NAT Configuration

If you use Network Address Translation (NAT), you should configure static NAT translations so that IPsec works properly. In general, NAT translation should occur before the router performs IPsec encapsulation; in other words, IPsec should be working with global addresses.

Nested IPsec Tunnels

Cisco IOS XE IPsec supports nested tunnels that terminate on the same router. Double encryption of locally generated IKE packets and IPsec packets is supported only when a static virtual tunnel interface (sVTI) is configured. Double encryption is not supported on later releases.

IPv4 Packets

Cisco IOS XE does not support IPv4 options; hence, the crypto engine drops IPv4 packets that contain IPv4 options.

Access Control Lists

Cisco IOS XE does not support access control lists (ACLs) that have discontiguous masks in IPsec.

Information About Configuring Security for VPNs with IPsec

Supported Standards

Cisco implements the following standards with this feature:

- IPsec--IP Security Protocol. IPsec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPsec provides these security services at the IP layer; it uses IKE to handle negotiation of protocols and algorithms based on local policy, and to generate the encryption and authentication keys to be used by IPsec. IPsec can be used to protect one or more data flows between a pair of hosts, between a pair of security gateways, or between a security gateway and a host.
- IKE--A hybrid protocol that implements Oakley and SKEME key exchanges inside the ISAKMP framework. While IKE can be used with other protocols, its initial implementation is with the IPsec protocol. IKE provides authentication of the IPsec peers, negotiates IPsec security associations, and establishes IPsec keys.

Note The term IPsec is sometimes used to describe the entire protocol of IPsec data services and IKE security protocols and is also sometimes used to describe only the data services.

The component technologies implemented for IPsec include:

- AES--Advanced Encryption Standard. A cryptographic algorithm that protects sensitive, unclassified information. AES is privacy transform for IPsec and IKE and has been developed to replace the DES. AES is designed to be more secure than DES. AES offers a larger key size while ensuring that the only known approach to decrypt a message is for an intruder to try every possible key. AES has a variable key length--the algorithm can specify a 128-bit key (the default), a 192-bit key, or a 256-bit key.
- DES--Data Encryption Standard. An algorithm that is used to encrypt packet data. Cisco IOS XE implements the mandatory 56-bit DES-CBC with Explicit IV. Cipher Block Chaining (CBC) requires an initialization vector (IV) to start encryption. The IV is explicitly given in the IPsec packet. For backwards compatibility, Cisco IOS XE IPsec also implements the RFC 1829 version of ESP DES-CBC.

Cisco IOS XE also implements Triple DES (168-bit) encryption, depending on the software versions available for a specific platform. Triple DES (3DES) is a strong form of encryption that allows sensitive information to be transmitted over untrusted networks. It enables customers to utilize network layer encryption.

Note Cisco IOS XE images with strong encryption (including, but not limited to, 56-bit data encryption feature sets) are subject to United States government export controls, and have a limited distribution. Images to be installed outside the United States require an export license. Customer orders might be denied or subject to delay due to United States government regulations. Contact your sales representative or distributor for more information or e-mail export@cisco.com.

- MD5 (HMAC variant)--MD5 (Message Digest 5) is a hash algorithm. HMAC is a keyed hash variant used to authenticate data.
- SHA (HMAC variant)--SHA (Secure Hash Algorithm) is a hash algorithm. HMAC is a keyed hash variant used to authenticate data.

IPsec as implemented in the Cisco IOS XE software supports the following additional standards:

- AH--Authentication Header. A security protocol which provides data authentication and optional anti-replay services. AH is embedded in the data to be protected (a full IP datagram).
- ESP--Encapsulating Security Payload. A security protocol which provides data privacy services and optional data authentication, and anti-replay services. ESP encapsulates the data to be protected.

Supported Hardware Switching Paths and Encapsulation

Supported Switching Paths

Table 1 Supported Switching Paths for IPsec

Switching Paths	Examples

Process switching	`interface fastethernet0/0/1`
	` no ip route-cache`

Fast switching	`interface fastethernet0/0/1`
	` ip route-cache`
	`! Ensure that you will not hit flow`
	`switching.`
	` no ip route-cache flow`
	`! Disable CEF for the interface, which`
	`supercedes global CEF.`
	` no ip route-cache cef`

Cisco Express Forwarding (CEF)	`ip cef`
	`interface fastethernet0/0/1`
	` ip route-cache`
	`! Ensure that you will not hit flow`
	`switching.`
	` no ip route-cache flow`

Fast-flow switching	`interface fastethernet0/0/1`
	` ip route-cache`
	`! Enable flow switching`
	`p route-cache flow`
	`! Disable CEF for the interface.`
	`  no ip route-cache cef`

CEF-flow switching	`! Enable global CEF.`
	`ip cef`
	`interface fastethernet0/0/1`
	` ip route-cache`
	` ip route-cache flow`
	`! Enable CEF for the interface`
	` ip route-cache cef`

Supported Encapsulation

IPsec works with the following serial encapsulations: High-Level Data-Links Control (HDLC), PPP, and Frame Relay.

IPsec also works with the Generic Routing Encapsulation (GRE) and IPinIP Layer 3 protocols. Other Layer 3 tunneling protocols may not be supported for use with IPsec.

Because the IPsec Working Group has not yet addressed the issue of group key distribution, IPsec currently cannot be used to protect group traffic (such as broadcast or multicast traffic).

IPsec Functionality Overview

IPsec provides the following network security services. (In general, local security policy dictates the use of one or more of these services.)

- Data Confidentiality--The IPsec sender can encrypt packets before transmitting them across a network.
- Data Integrity--The IPsec receiver can authenticate packets sent by the IPsec sender to ensure that the data has not been altered during transmission.
- Data Origin Authentication--The IPsec receiver can authenticate the source of the IPsec packets sent. This service is dependent upon the data integrity service.
- Anti-Replay--The IPsec receiver can detect and reject replayed packets.

IPsec provides secure tunnels between two peers, such as two routers. You define which packets are considered sensitive and should be sent through these secure tunnels, and you define the parameters that should be used to protect these sensitive packets, by specifying characteristics of these tunnels. Then, when the IPsec peer recognizes such a sensitive packet, the peer sets up the appropriate secure tunnel and sends the packet through the tunnel to the remote peer. (The use of the term tunnel in this chapter does not refer to using IPsec in tunnel mode.)

More accurately, these tunnels are sets of security associations (SAs) that are established between two IPsec peers. The SAs define which protocols and algorithms should be applied to sensitive packets and specify the keying material to be used by the two peers. SAs are unidirectional and are established per security protocol (AH or ESP).

With IPsec, you define what traffic should be protected between two IPsec peers by configuring access lists and applying these access lists to interfaces by way of crypto map sets. Therefore, traffic may be selected on the basis of source and destination address, and optionally Layer 4 protocol, and port. (The access lists used for IPsec are used only to determine which traffic should be protected by IPsec, and not which traffic should be blocked or permitted through the interface. Separate access lists define blocking and permitting at the interface.)

A crypto map set can contain multiple entries, each with a different access list. The crypto map entries are searched in order--the router attempts to match the packet to the access list specified in that entry.

When a packet matches a **permit** entry in a particular access list, and the corresponding crypto map entry is tagged as **cisco**, connections are established, if necessary. If the crypto map entry is tagged as **ipsec-isakmp**, IPsec is triggered. If no SA exists that IPsec can use to protect this traffic to the peer, IPsec uses IKE to negotiate with the remote peer to set up the necessary IPsec SAs on behalf of the data flow. The negotiation uses information specified in the crypto map entry as well as the data flow information from the specific access list entry. (The behavior is different for dynamic crypto map entries. See the “Creating Dynamic Crypto Maps, page 20” section later in this module.)

If the crypto map entry is tagged as **ipsec-manual**, IPsec is triggered. If no SA exists that IPsec can use to protect this traffic to the peer, the traffic is dropped. In this case, the SAs are installed via the configuration, without the intervention of IKE. If the SAs did not exist, IPsec did not have all of the necessary pieces configured.

Once established, the set of SAs (outbound, to the peer) is then applied to the triggering packet and to subsequent applicable packets as those packets exit the router. “Applicable” packets are packets that match the same access list criteria that the original packet matched. For example, all applicable packets could be encrypted before being forwarded to the remote peer. The corresponding inbound SAs are used when processing the incoming traffic from that peer.

Multiple IPsec tunnels can exist between two peers to secure different data streams, with each tunnel using a separate set of SAs. For example, some data streams might be authenticated only while other data streams must both be encrypted and authenticated.

Access lists associated with IPsec crypto map entries also represent which traffic the router requires to be protected by IPsec. Inbound traffic is processed against the crypto map entries--if an unprotected packet matches a **permit** entry in a particular access list associated with an IPsec crypto map entry, that packet is dropped because it was not sent as an IPsec-protected packet.

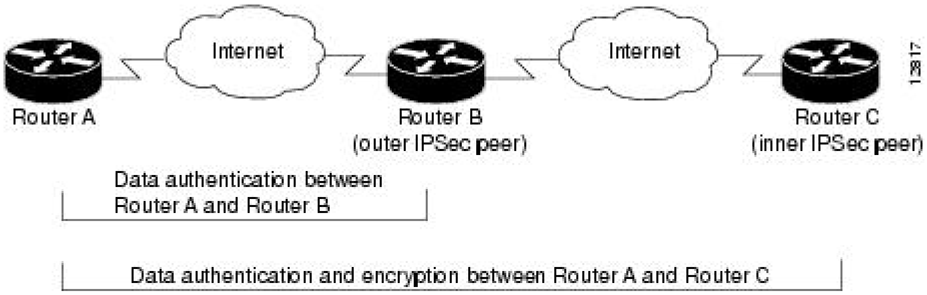
Crypto map entries also include transform sets. A transform set is an acceptable combination of security protocols, algorithms, and other settings to apply to IPsec protected traffic. During the IPsec SA negotiation, the peers agree to use a particular transform set when protecting a particular data flow.

IPsec Traffic Nested to Multiple Peers

You can nest IPsec traffic to a series of IPsec peers. For example, in order for traffic to traverse multiple firewalls (these firewalls have a policy of not letting through traffic that they have not authenticated), the router must establish IPsec tunnels with each firewall in turn. The “nearer” firewall becomes the “outer” IPsec peer.

In the example shown in the figure below, Router A encapsulates the traffic destined for Router C in IPsec (Router C is the inner IPsec peer). However, before Router A can send this traffic, it must first reencapsulate this traffic in IPsec in order to send it to Router B (Router B is the “outer” IPsec peer).

Figure 1 Nesting Example of IPsec Peers



It is possible for the traffic between the “outer” peers to have one kind of protection (such as data authentication) and for traffic between the “inner” peers to have different protection (such as both data authentication and encryption).

Crypto Access Lists

- Crypto Access List Overview, page 6
- When to Use the permit and deny Keywords in Crypto Access Lists, page 7
- Mirror Image Crypto Access Lists at Each IPsec Peer, page 9
- When to Use the any Keyword in Crypto Access Lists, page 10

Crypto Access List Overview

Crypto access lists are used to define which IP traffic is protected by crypto and which traffic is not protected by crypto. (These access lists are not the same as regular access lists, which determine what traffic to forward or block at an interface.) For example, access lists can be created to protect all IP traffic between Subnet A and Subnet Y or Telnet traffic between Host A and Host B.

The access lists themselves are not specific to IPsec. It is the crypto map entry referencing the specific access list that defines whether IPsec processing is applied to the traffic matching a **permit** in the access list.

Crypto access lists associated with IPsec crypto map entries have the following primary functions:

- Select outbound traffic to be protected by IPsec (permit = protect).
- Indicate that the data flow to be protected by the new SAs (specified by a single **permit** entry) when initiating negotiations for IPsec security associations.
- Process inbound traffic in order to filter out and discard traffic that should have been protected by IPsec.
- Determine whether or not to accept requests for IPsec security associations on behalf of the requested data flows when processing IKE negotiation from the IPsec peer.
- Negotiation is performed only for **ipsec-isakmp** crypto map entries. In order to be accepted, if the peer initiates the IPsec negotiation, it must specify a data flow that is "permitted" by a crypto access list associated with an **ipsec-isakmp** crypto map entry.

If you want certain traffic to receive one combination of IPsec protection (for example, authentication only) and other traffic to receive a different combination of IPsec protection (for example, both authentication and encryption), you need to create two different crypto access lists to define the two different types of traffic. These different access lists are then used in different crypto map entries which specify different IPsec policies.

When to Use the permit and deny Keywords in Crypto Access Lists

Crypto protection can be permitted or denied for certain IP traffic in a crypto access list as follows:

- To protect IP traffic that matches the specified policy conditions in its corresponding crypto map entry, use the **permit** keyword in an access list.
- To refuse protection for IP traffic that matches the specified policy conditions in its corresponding crypto map entry, use the **deny** keyword in an access list.

Note IP traffic is not protected by crypto if it is refused protection in all of the crypto map entries for an interface.

After the corresponding crypto map entry is defined and the crypto map set is applied to the interface, the defined crypto access list is applied to an interface. Different access lists must be used in different entries of the same crypto map set. However, both inbound and outbound traffic is evaluated against the same "outbound" IPsec access list. Therefore, the access list's criteria is applied in the forward direction to traffic exiting your router and in the reverse direction to traffic entering your router.

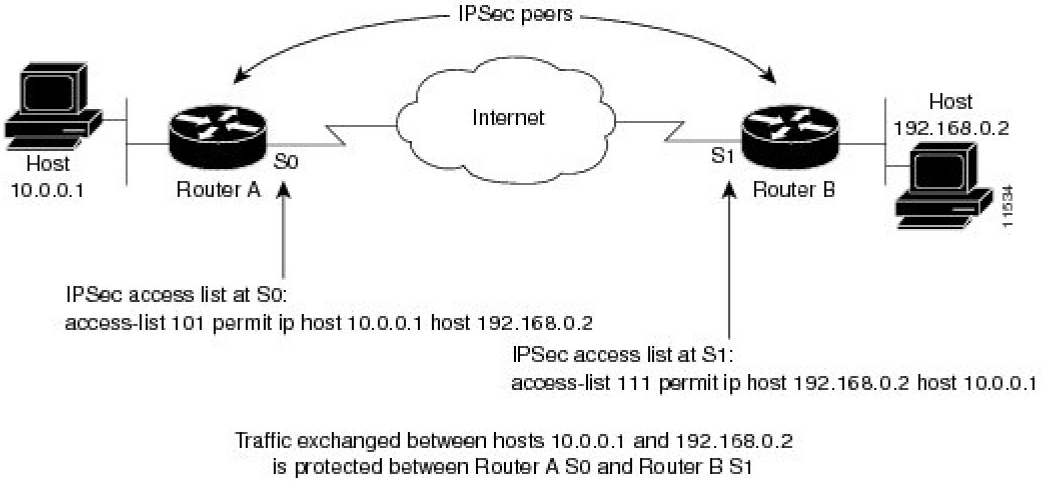
In the figure below, IPsec protection is applied to traffic between Host 10.0.0.1 and Host 192.168.0.2 as the data exits Router A's S0 interface en route to Host 192.168.0.2. For traffic from Host 10.0.0.1 to Host 192.168.0.2, the access list entry on Router A is evaluated as follows:

```
source = host 10.0.0.1
dest = host 192.168.0.2
```

For traffic from Host 192.168.0.2 to Host 10.0.0.1, that same access list entry on Router A is evaluated as follows:

```
source = host 192.168.0.2
dest = host 10.0.0.1
```

Figure 2 How Crypto Access Lists Are Applied for Processing IPsec


IPSec peers
Internet
Host
10.0.0.1
Router A
S0
S1
Router B
Host
192.168.0.2
11534
IPSec access list at S0:
access-list 101 permit ip host 10.0.0.1 host 192.168.0.2
IPSec access list at S1:
access-list 111 permit ip host 192.168.0.2 host 10.0.0.1
Traffic exchanged between hosts 10.0.0.1 and 192.168.0.2
is protected between Router A S0 and Router B S1

If you configure multiple statements for a given crypto access list that is used for IPsec, in general the first **permit** statement that is matched is the statement used to determine the scope of the IPsec SA. That is, the IPsec SA is set up to protect traffic that meets the criteria of the matched statement only. Later, if traffic matches a different **permit** statement of the crypto access list, a new, separate IPsec SA is negotiated to protect traffic matching the newly matched access list statement.

Any unprotected inbound traffic that matches a **permit** entry in the crypto access list for a crypto map entry flagged as IPsec is dropped, because this traffic was expected to be protected by IPsec.

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

The following example shows that if overlapping networks are used, then the most specific networks are defined in crypto sequence numbers before less specific networks are defined. In this example, the more specific network is covered by the crypto map sequence number 10, followed by the less specific network in the crypto map, which is sequence number 20.

```
crypto map mymap 10 ipsec-isakmp
 set peer 192.168.1.1
 set transform-set test
 match address 101
crypto map mymap 20 ipsec-isakmp
 set peer 192.168.1.2
 set transform-set test
 match address 102
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 102 permit ip 10.0.0.0 0.255.255.255 172.16.0.0 0.15.255.255
```

The following example shows how having the **deny** keyword in one crypto map sequence number and having the **permit** keyword for the same subnet and IP range in another crypto map sequence number is not supported.

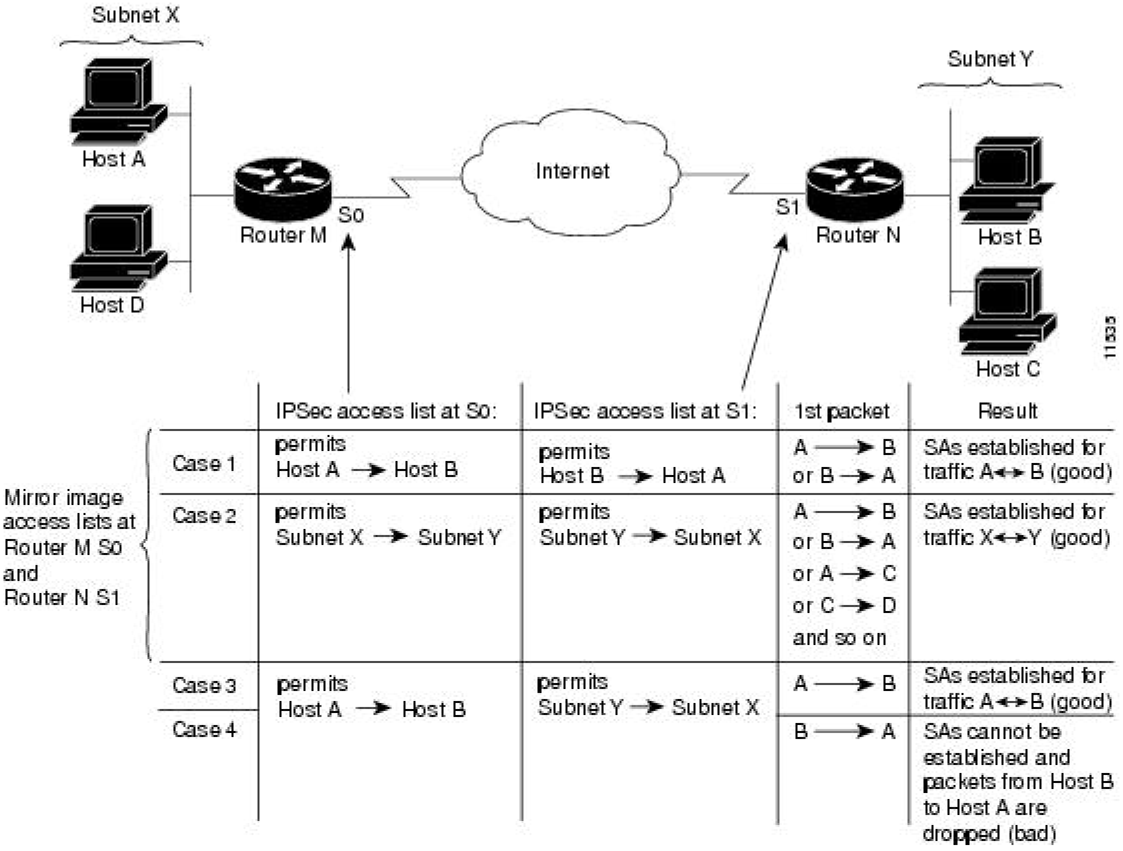
```
crypto map mymap 10 ipsec-isakmp
 set peer 192.168.1.1
 set transform-set test
 match address 101
crypto map mymap 20 ipsec-isakmp
 set peer 192.168.1.2
 set transform-set test
 match address 102
access-list 101 deny   ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 101 permit ip 10.0.0.0 0.255.255.255 172.16.0.0 0.15.255.255
access-list 102 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
```

Mirror Image Crypto Access Lists at Each IPsec Peer

Cisco recommends that for every crypto access list specified for a static crypto map entry that you define at the local peer, you define a “mirror image” crypto access list at the remote peer. This ensures that traffic that has IPsec protection applied locally can be processed correctly at the remote peer. (The crypto map entries themselves must also support common transforms and must refer to the other system as a peer.)

The figure below shows some sample scenarios when you have mirror image access lists and when you do not have mirror image access lists.

Figure 3 Mirror Image vs. Nonmirror Image Crypto Access Lists (for IPsec)



		IPSec access list at S0:	IPSec access list at S1:	1st packet	Result
Mirror image access lists at Router M So and Router N S1	Case 1	permits Host A → Host B	permits Host B → Host A	A → B or B → A	SAs established for traffic A↔ B (good)
	Case 2	permits Subnet X → Subnet Y	permits Subnet Y → Subnet X	A → B or B → A or A → C or C → D and so on	SAs established for traffic X↔Y (good)
	Case 3	permits Host A → Host B	permits Subnet Y → Subnet X	A → B	SAs established for traffic A↔B (good)
	Case 4			B → A	SAs cannot be established and packets from Host B to Host A are dropped (bad)

As the figure above indicates, IPsec SAs can be established as expected whenever the two peers' crypto access lists are mirror images of each other. However, an IPsec SA can be established only some of the time when the access lists are not mirror images of each other. This can happen in the case where an entry in one peer's access list is a subset of an entry in the other peer's access list, such as shown in Cases 3 and 4 of the figure. IPsec SA establishment is critical to IPsec--without SAs, IPsec does not work, causing any packets matching the crypto access list criteria to be silently dropped instead of being forwarded with IPsec.

In the figure above, an SA cannot be established in Case 4. This is because SAs are always requested according to the crypto access lists at the initiating packet's end. In Case 4, Router N requests that all traffic between Subnet X and Subnet Y be protected, but this is a superset of the specific flows permitted by the crypto access list at Router M. Therefore, the request is not permitted. Case 3 works because Router M's request is a subset of the specific flows permitted by the crypto access list at Router N.

Because of the complexities introduced when crypto access lists are not configured as mirror images at peer IPsec devices, Cisco strongly encourages you to use mirror image crypto access lists.

When to Use the any Keyword in Crypto Access Lists

When you create crypto access lists, using the **any** keyword could cause problems. Cisco discourages the use of the **any** keyword to specify source or destination addresses.

The **any** keyword in a **permit** statement is discouraged when you have multicast traffic flowing through the IPsec interface; the **any** keyword can cause multicast traffic to fail.

The **permit any any** statement is strongly discouraged, because this causes all outbound traffic to be protected (and all protected traffic sent to the peer specified in the corresponding crypto map entry) and requires protection for all inbound traffic. Then, all inbound packets that lack IPsec protection are silently dropped, including packets for routing protocols, Network Time Protocol (NTP), echo, echo response, and so on.

You need to be sure you define which packets to protect. If you must use the **any** keyword in a **permit** statement, you must preface that statement with a series of **deny** statements to filter out any traffic (that would otherwise fall within that **permit** statement) that you do not want to be protected.

Also, use of the **any** keyword in access control lists (ACLs) with reverse route injection (RRI) is not supported. (For more information about RRI, see the section "Creating Crypto Map Sets, page 18.")

Transform Sets: A Combination of Security Protocols and Algorithms

- About Transform Sets, page 10

About Transform Sets

A transform set represents a certain combination of security protocols and algorithms. During the IPsec SA negotiation, the peers agree to use a particular transform set for protecting a particular data flow.

You can specify multiple transform sets, and then specify one or more of these transform sets in a crypto map entry. The transform set defined in the crypto map entry is used in the IPsec SA negotiation to protect the data flows specified by that crypto map entry's access list.

During IPsec security association negotiations with IKE, the peers search for a transform set that is the same at both peers. When such a transform set is found, it is selected and is applied to the protected traffic as part of both peers' IPsec SAs. (With manually established SAs, there is no negotiation with the peer, so both sides must specify the same transform set.)

If you change a transform set definition, the change is only applied to crypto map entries that reference the transform set. The change is not applied to existing security associations, but is used in subsequent negotiations to establish new SAs. If you want the new settings to take effect sooner, you can clear all or part of the SA database by using the **clear crypto sa** command.

The table below shows allowed transform combinations.

Table 2 Allowed Transform Combinations

Transform Type	Transform	Description
AH Transform	**ah-md5-hmac** **ah-sha-hmac**	AH with the MD5 (a Hash-based Message Authentication Code [HMAC] variant) authentication algorithm AH with the SHA (an HMAC variant) authentication algorithm
ESP Encryption Transform	**esp-aes** **esp-aes 192** **esp-aes 256** **esp-des** **esp-3des esp-null**	ESP with the 128-bit AES encryption algorithm ESP with the 192-bit AES encryption algorithm ESP with the 256-bit AES encryption algorithm ESP with the 56-bit DES encryption algorithm ESP with the 168-bit DES encryption algorithm (3DES or Triple DES) Null encryption algorithm
ESP Authentication Transform	**esp-md5-hmac** **esp-sha-hmac**	ESP with the MD5 (HMAC variant) authentication algorithm ESP with the SHA (HMAC variant) authentication algorithm

Crypto Map Sets

Before you create crypto map entries, you should determine which type of crypto map--static, dynamic, or manual--best addresses the needs of your network.

- About Crypto Maps, page 12
- Load Sharing Among Crypto Maps, page 12
- Crypto Map Guidelines, page 13
- Dynamic Crypto Maps, page 13
- Redundant Interfaces Sharing the Same Crypto Map, page 14
- Establishing Manual SAs, page 14

About Crypto Maps

Crypto map entries created for IPsec pull together the various parts used to set up IPsec SAs, including:

- Which traffic should be protected by IPsec (per a crypto access list)
- The granularity of the flow to be protected by a set of SAs
- Where IPsec-protected traffic should be sent (who the remote IPsec peer is)
- The local address to be used for the IPsec traffic (See the section “Applying Crypto Map Sets to Interfaces, page 26” for more details.)
- What IPsec SA should be applied to this traffic (selecting from a list of one or more transform sets)
- Whether SAs are manually established or are established via IKE
- Other parameters that might be necessary to define an IPsec SA

How Crypto Maps Work

Crypto map entries with the same crypto map name (but different map sequence numbers) are grouped into a crypto map set. Later, you apply these crypto map sets to interfaces; then, all IP traffic passing through the interface is evaluated against the applied crypto map set. If a crypto map entry sees outbound IP traffic that should be protected and the crypto map specifies the use of IKE, an SA is negotiated with the remote peer according to the parameters included in the crypto map entry; otherwise, if the crypto map entry specifies the use of manual SAs, an SA should have already been established via configuration. (If a dynamic crypto map entry sees outbound traffic that should be protected and no security association exists, the packet is dropped.)

The policy described in the crypto map entries is used during the negotiation of SAs. If the local router initiates the negotiation, it uses the policy specified in the static crypto map entries to create the offer to be sent to the specified IPsec peer. If the IPsec peer initiates the negotiation, the local router checks the policy from the static crypto map entries, and any referenced dynamic crypto map entries to decide whether to accept or reject the peer’s request (offer).

For IPsec to succeed between two IPsec peers, both peers’ crypto map entries must contain compatible configuration statements.

Compatible Crypto Maps: Establishing an SA

When two peers try to establish an SA, they must each have at least one crypto map entry that is compatible with one of the other peer’s crypto map entries. For two crypto map entries to be compatible, they must at least meet the following criteria:

- The crypto map entries must contain compatible crypto access lists (for example, mirror image access lists). In the case where the responding peer is using dynamic crypto maps, the entries in the local crypto access list must be “permitted” by the peer’s crypto access list.
- The crypto map entries must each identify the other peer (unless the responding peer is using dynamic crypto maps).
- The crypto map entries must have at least one transform set in common.

Load Sharing Among Crypto Maps

You can define multiple remote peers using crypto maps to allow for load sharing. Load sharing is useful because if one peer fails, there continues to be a protected path. The peer that packets are actually sent to is determined by the last peer that the router heard from (received either traffic or a negotiation request from) for a given data flow. If the attempt fails with the first peer, IKE tries the next peer on the crypto map list.

If you are not sure how to configure each crypto map parameter to guarantee compatibility with other peers, you might consider configuring dynamic crypto maps as described in the section "Creating Dynamic Crypto Maps, page 20." Dynamic crypto maps are useful when the establishment of the IPsec tunnels is initiated by the remote peer (such as in the case of an IPsec router fronting a server). They are not useful if the establishment of the IPsec tunnels is locally initiated, because the dynamic crypto maps are policy templates, not complete statements of policy. (Although the access lists in any referenced dynamic crypto map entry are used for crypto packet filtering.)

Crypto Map Guidelines

You can apply only one crypto map set to a single interface. The crypto map set can include a combination of IPsec/IKE and IPsec/manual entries. Multiple interfaces can share the same crypto map set if you want to apply the same policy to multiple interfaces.

If you create more than one crypto map entry for a given interface, use the *seq-num* argument of each map entry to rank the map entries: the lower the *seq-num* argument, the higher the priority. At the interface that has the crypto map set, traffic is evaluated against higher priority map entries first.

You must create multiple crypto map entries for a given interface if any of the following conditions exist:

- If different data flows are to be handled by separate IPsec peers.
- If you want to apply different IPsec security to different types of traffic (to the same or separate IPsec peers); for example, if you want traffic between one set of subnets to be authenticated, and traffic between another set of subnets to be both authenticated and encrypted. In this case the different types of traffic should have been defined in two separate access lists, and you must create a separate crypto map entry for each crypto access list.

If you are not using IKE to establish a particular set of security associations, and want to specify multiple access list entries, you must create separate access lists (one per **permit** entry) and specify a separate crypto map entry for each access list.

Dynamic Crypto Maps

Dynamic crypto maps can ease IPsec configuration and are recommended for use with networks where the peers are not always predetermined.

Note Dynamic crypto maps are only available for use by IKE.

A dynamic crypto map entry is essentially a static crypto map entry without all the parameters configured. It acts as a policy template where the missing parameters are later dynamically configured (as the result of an IPsec negotiation) to match a remote peer's requirements. This allows remote peers to exchange IPsec traffic with the router even if the router does not have a crypto map entry specifically configured to meet all of the remote peer's requirements.

Dynamic crypto maps are not used by the router to initiate new IPsec security associations with remote peers. Dynamic crypto maps are used when a remote peer tries to initiate an IPsec security association with the router. Dynamic crypto maps are also used in evaluating traffic.

A dynamic crypto map set is included by reference as part of a crypto map set. Any crypto map entries that reference dynamic crypto map sets should be the lowest priority crypto map entries in the crypto map set (that is, have the highest sequence numbers) so that the other crypto map entries are evaluated first; that way, the dynamic crypto map set is examined only when the other (static) map entries are not successfully matched.

If the router accepts the peer's request, at the point that it installs the new IPsec security associations it also installs a temporary crypto map entry. This entry is filled in with the results of the negotiation. At this point, the router performs normal processing, using this temporary crypto map entry as a normal entry, even requesting new security associations if the current ones are expiring (based upon the policy specified in the temporary crypto map entry). Once the flow expires (that is, all of the corresponding security associations expire), the temporary crypto map entry is then removed.

For both static and dynamic crypto maps, if unprotected inbound traffic matches a **permit** statement in an access list, and the corresponding crypto map entry is tagged as "IPsec," then the traffic is dropped because it is not IPsec-protected. (This is because the security policy as specified by the crypto map entry states that this traffic must be IPsec-protected.)

For static crypto map entries, if outbound traffic matches a **permit** statement in an access list and the corresponding SA is not yet established, the router initiates new SAs with the remote peer. In the case of dynamic crypto map entries, if no SA existed, the traffic would simply be dropped (because dynamic crypto maps are not used for initiating new SAs).

Note Be careful when using the **any** keyword in **permit** entries in dynamic crypto maps. If it is possible for the traffic covered by such a **permit** entry to include multicast or broadcast traffic, the access list should include **deny** entries for the appropriate address range. Access lists should also include **deny** entries for network and subnet broadcast traffic, and for any other traffic that should not be IPsec protected.

Redundant Interfaces Sharing the Same Crypto Map

For redundancy, you could apply the same crypto map set to more than one interface. The default behavior is as follows:

- Each interface has its own piece of the security association database.
- The IP address of the local interface is used as the local address for IPsec traffic originating from or destined to that interface.

If you apply the same crypto map set to multiple interfaces for redundancy purposes, you need to specify an identifying interface. One suggestion is to use a loopback interface as the identifying interface. This has the following effects:

- The per-interface portion of the IPsec security association database is established one time and shared for traffic through all the interfaces that share the same crypto map.
- The IP address of the identifying interface is used as the local address for IPsec traffic originating from or destined to those interfaces sharing the same crypto map set.

Establishing Manual SAs

The use of manual security associations is a result of a prior arrangement between the users of the local router and the IPsec peer. The two parties may begin with manual SAs and then move to using SAs established via IKE, or the remote party's system may not support IKE. If IKE is not used for establishing the SAs, there is no negotiation of SAs, so the configuration information in both systems must be the same in order for traffic to be processed successfully by IPsec.

The local router can simultaneously support manual and IKE-established SAs, even within a single crypto map set.

There is very little reason to disable IKE on the local router (unless the router only supports manual SAs, which is unlikely).

Note Access lists for crypto map entries tagged as **ipsec-manual** are restricted to a single **permit** entry and subsequent entries are ignored. In other words, the SAs established by that particular crypto map entry are only for a single data flow. To be able to support multiple manually established SAs for different kinds of traffic, define multiple crypto access lists, and then apply each one to a separate **ipsec-manual** crypto map entry. Each access list should include one **permit** statement defining what traffic to protect.

How to Configure IPsec VPNs

Creating Crypto Access Lists

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. Do one of the following:
    - **access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**log**]
    - **ip access-list extended** *name*
4. Repeat Step 3 for each crypto access list that you want to create.

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose

Step 3	Do one of the following:
	• **access-list** *access-list-number* {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**log**]
	• **ip access-list extended** *name*
	Example:
	`Router(config)# access-list 100 permit ip 10.0.68.0 0.0.0.255 10.1.1.0 0.0.0.255`
	Example:
	`Router(config)# ip access-list extended vpn-tunnel`
		Specifies conditions to determine which IP packets are protected. You specify conditions using an IP access list designated by either a number or a name. The **access-list** command designates a numbered extended access list; the **ip access-list extended** command designates a named access list.
		Enable or disable crypto for traffic that matches these conditions.
		Tip Cisco recommends that you configure "mirror image" crypto access lists for use by IPsec and that you avoid using the **any** keyword.

Step 4	Repeat Step 3 for each crypto access list that you want to create.	--

- What to Do Next, page 16

What to Do Next

After at least one crypto access list is created, a transform set needs to be defined as described in the section "Defining Transform Sets, page 16."

Next the crypto access lists need to be associated to particular interfaces when you configure and apply crypto map sets to the interfaces are configured and applied (following instructions in the sections "Creating Crypto Map Sets, page 18" and "Applying Crypto Map Sets to Interfaces, page 26").

Defining Transform Sets

Perform this task to define a transform set that is to be used by the IPsec peers during IPsec security association negotiations with IKE.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec transform-set** *transform-set-name transform1* [*transform2* [*transform3*]]
4. **mode** [**tunnel** | **transport**]
5. **exit**
6. **clear crypto sa** [**peer** {*ip-address* | *peer-name*} | **sa map** *map-name* | **sa entry** *destination-address protocol spi*]
7. **show crypto ipsec transform-set** [**tag** *transform-set-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. - Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec transform-set** *transform-set-name* *transform1* [*transform2* [*transform3*]] **Example:** `Router(config)# crypto ipsec transform-set aesset esp-aes 256 esp-sha-hmac`	Defines a transform set and enters crypto transform configuration mode. There are complex rules defining which entries you can use for the transform arguments. These rules are explained in the command description for the **crypto ipsec transform-set** command, and the table in the About Transform Sets, page 10 section provides a list of allowed transform combinations.
Step 4	**mode** [**tunnel** | **transport**] **Example:** `Router(cfg-crypto-tran)# mode transport`	(Optional) Changes the mode associated with the transform set. The mode setting is applicable only to traffic whose source and destination addresses are the IPsec peer addresses; it is ignored for all other traffic. (All other traffic is in tunnel mode only.)
Step 5	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 6	**clear crypto sa** [**peer** {*ip-address* | *peer-name*} | **sa map** *map-name* | **sa entry** *destination-address protocol spi*] **Example:** `Router# clear crypto sa`	(Optional) Clears existing IPsec security associations so that any changes to a transform set takes effect on subsequently established security associations. Manually established SAs are reestablished immediately. - Using the **clear crypto sa** command without parameters clear out the full SA database, which clears out active security sessions. - You may also specify the **peer**, **map**, or **entry** keywords to clear out only a subset of the SA database.
Step 7	**show crypto ipsec transform-set** [**tag** *transform-set-name*] **Example:** `Router# show crypto ipsec transform-set`	(Optional) Displays the configured transform sets.

- What to Do Next, page 18

What to Do Next

After you have defined a transform set, you should create a crypto map as specified in the Creating Crypto Map Sets section.

Creating Crypto Map Sets

- Creating Static Crypto Maps, page 18
- Creating Dynamic Crypto Maps, page 20
- Creating Crypto Map Entries to Establish Manual SAs, page 23

Creating Static Crypto Maps

When IKE is used to establish SAs, the IPsec peers can negotiate the settings they use for the new security associations. This means that you can specify lists (such as lists of acceptable transforms) within the crypto map entry.

Perform this task to create crypto map entries that use IKE to establish the SAs.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto map** *map-name seq-num* **ipsec-isakmp**
4. **match address** *access-list-id*
5. **set peer** {*hostname* | *ip-address*}
6. **set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*]
7. **set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*}
8. **set security-association level per-host**
9. **set pfs** [**group1** | **group2** | **group5**]
10. **end**
11. **show crypto map** [**interface** *interface* | **tag** *map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

Step 3	**crypto map** *map-name seq-num* **ipsec-isakmp**
	Example:
	`Router(config)# crypto map static-map 1`
	`ipsec-isakmp`
		Names the crypto map entry to create (or modify) and enters crypto map configuration mode.

Step 4	**match address** *access-list-id*
	Example:
	`Router(config-crypto-m)# match address`
	`vpn-tunnel`
		Names an extended access list.
		This access list determines which traffic should be protected by IPsec and which traffic should not be protected by IPsec security in the context of this crypto map entry.

Step 5	**set peer** {*hostname* | *ip-address*}
	Example:
	`Router(config-crypto-m)# set-peer`
	`192.168.101.1`
		Specifies a remote IPsec peer, the peer to which IPsec protected traffic can be forwarded.
		Repeat for multiple remote peers.

Step 6	**set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*]
	Example:
	`Router(config-crypto-m)# set transform-`
	`set aesset`
		Specifies which transform sets are allowed for this crypto map entry.
		List multiple transform sets in order of priority (highest priority first).

Step 7	**set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*}
	Example:
	`Router (config-crypto-m)# set security-`
	`association lifetime seconds 2700`
		(Optional) Specifies an SA lifetime for the crypto map entry.
		By default, the SAs of the crypto map are negotiated according to the global lifetimes.

Step 8	**set security-association level per-host**
	Example:
	`Router(config-crypto-m)# set security-`
	`association level per-host`
		(Optional) Specifies that separate SAs should be established for each source and destination host pair.
		By default, a single IPsec "tunnel" can carry traffic for multiple source hosts and multiple destination hosts.
		Caution Use this command with care, because multiple streams between given subnets can rapidly consume resources.

	Command or Action	Purpose
Step 9	**set pfs** [**group1** | **group2** | **group5**] **Example:** `Router(config-crypto-m)# set pfs group2`	(Optional) Specifies that IPsec either should ask for perfect forward secrecy (PFS) when requesting new SAs for this crypto map entry or should demand PFS in requests received from the IPsec peer. By default, PFS is not requested. If no group is specified with this command, group1 is used as the default.

Step 10	**end**
	Example:
	`Router(config-crypto-m)# end`
		Exits crypto-map configuration mode and returns to privileged EXEC mode.

Step 11	**show crypto map** [**interface** *interface* | **tag** *map-name*] **Example:** `Router# show crypto map`	Displays your crypto map configuration.

- Troubleshooting Tips, page 20
- What to Do Next, page 20

Troubleshooting Tips

Certain configuration changes take effect only when negotiating subsequent SAs. If you want the new settings to take immediate effect, you must clear the existing SAs so that they are reestablished with the changed configuration. If the router is actively processing IPsec traffic, it is desirable to clear only the portion of the SA database that would be affected by the configuration changes (that is, clear only the SAs established by a given crypto map set). Clearing the full SA database should be reserved for large-scale changes, or when the router is processing very little other IPsec traffic.

To clear IPsec SAs, use the **clear crypto sa** command with appropriate parameters. (Omitting all parameters clears out the full SA database, which clears active security sessions.)

What to Do Next

After you have successfully created a static crypto map, you must apply the crypto map set to each interface through which IPsec traffic flows. To complete this task, see the section “Applying Crypto Map Sets to Interfaces, page 26.”

Creating Dynamic Crypto Maps

Perform this task to create dynamic crypto map entries that use IKE to establish the SAs.

Note Dynamic crypto map entries specify crypto access lists that limit traffic for which IPsec SAs can be established. A dynamic crypto map entry that does not specify an access list is ignored during traffic filtering. A dynamic crypto map entry with an empty access list causes traffic to be dropped. If there is only one dynamic crypto map entry in the crypto map set, it must specify acceptable transform sets.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto dynamic-map** *dynamic-map-name dynamic-seq-num*
4. **set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*]
5. **match address** *access-list-id*
6. **set peer** {*hostname* | *ip-address*}
7. **set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*}
8. **set pfs** [**group1**| **group2** | **group5**]
9. **end**
10. **show crypto dynamic-map** [**tag** *map-name*]
11. **configure terminal**
12. **crypto map** *map-name seq-num* **ipsec-isakmp dynamic** *dynamic-map-name*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto dynamic-map** *dynamic-map-name dynamic-seq-num* **Example:** `Router(config)# crypto dynamic-map test-map 1`	Creates a dynamic crypto map entry and enters crypto map configuration mode.

	Command or Action	Purpose
Step 4	**set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*] **Example:** `Router(config-crypto-m)# set transform-set aesset`	Specifies which transform sets are allowed for the crypto map entry. List multiple transform sets in order of priority (highest priority first). This is the only configuration statement required in dynamic crypto map entries.
Step 5	**match address** *access-list-id* **Example:** `Router(config-crypto-m)# match address 101`	(Optional) Accesses list number or name of an extended access list. This access list determines which traffic should be protected by IPsec and which traffic should not be protected by IPsec security in the context of this crypto map entry. **Note** Although access lists are optional for dynamic crypto maps, they are highly recommended. If access lists are configured, the data flow identity proposed by the IPsec peer must fall within a **permit** statement for this crypto access list. If access lists are not configured, the router accepts any data flow identity proposed by the IPsec peer. However, if access lists are configured but the specified access list does not exist or is empty, the router drops all packets. This scenario is similar to static crypto maps because they also require that an access list be specified. Care must be taken if the **any** keyword is used in the access list, because the access list is used for packet filtering as well as for negotiation. You must configure a match address; otherwise, the behavior is not secure.

Step 6	**set peer** {*hostname* | *ip-address*}
	Example:
	`Router(config-crypto-m)# set peer 192.168.101.1`
		(Optional) Specifies a remote IPsec peer. Repeat for multiple remote peers.
		Note This is rarely configured in dynamic crypto map entries. Dynamic crypto map entries are often used for unknown remote peers.

Step 7	**set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*}
	Example:
	`Router (config-crypto-m)# set security-association lifetime seconds 7200`
		(Optional) Overrides (for a particular crypto map entry) the global lifetime value, which is used when negotiating IPsec SAs.

Step 8	**set pfs** [**group1**| **group2** | **group5**] **Example:** `Router(config-crypto-m)# set pfs group2`	(Optional) Specifies that IPsec should ask for PFS when requesting new security associations for this crypto map entry or should demand PFS in requests received from the IPsec peer. By default, PFS is not requested. If no group is specified with this command, **group1** is used as the default.

	Command or Action	Purpose

Step 9	**end**
	Example:
	`Router(config-crypto-m)# end`
		Exits crypto-map configuration mode and returns to privileged EXEC mode.

Step 10	**show crypto dynamic-map** [**tag** *map-name*] **Example:** `Router# show crypto dynamic-map`	(Optional) Displays information about dynamic crypto maps.
Step 11	**configure terminal** **Example:** `Router# configure terminal`	Returns to global configuration mode.
Step 12	**crypto map** *map-name seq-num* **ipsec-isakmp dynamic** *dynamic-map-name* **Example:** `Router(config)# crypto map static-map 1 ipsec-isakmp dynamic test-map`	(Optional) Adds a dynamic crypto map to a crypto map set. You should set the crypto map entries referencing dynamic maps to the lowest priority entries in a crypto map set.

- Troubleshooting Tips, page 23
- What to Do Next, page 23

Troubleshooting Tips

Certain configuration changes take effect only when negotiating subsequent SAs. If you want the new settings to take immediate effect, you must clear the existing SAs so that they are reestablished with the changed configuration. If the router is actively processing IPsec traffic, it is desirable to clear only the portion of the SA database that would be affected by the configuration changes (that is, clear only the SAs established by a given crypto map set). Clearing the full SA database should be reserved for large-scale changes, or when the router is processing very little other IPsec traffic.

To clear IPsec SAs, use the **clear crypto sa** command with appropriate parameters. (Omitting all parameters clears the full SA database, which clears active security sessions.)

What to Do Next

After you have successfully created a crypto map set, you must apply the crypto map set to each interface through which IPsec traffic flows. To complete this task, see the section “Applying Crypto Map Sets to Interfaces, page 26.”

Creating Crypto Map Entries to Establish Manual SAs

To create crypto map entries to establish manual SAs (that is, when IKE is not used to establish the SAs), perform this task.

SUMMARY STEPS

1. **enable**
2. **configureterminal**
3. **crypto map** *map-name seq-num* **ipsec-manual**
4. **match address** *access-list-id*
5. **set peer** {*hostname* | *ip-address*}
6. **set transform-set** *transform-set-name*
7. Enter the following commands:
   - **set session-key inbound ah** *spi hex-key-string*
     and
   - **set session-key outbound ah** *spi hex-key-string*
8. Enter the following commands:
   - **set session-key inbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
     and
   - **set session-key outbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
9. **end**
10. **show crypto map interface** [*interface* | **tag** *map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configureterminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto map** *map-name seq-num* **ipsec-manual** **Example:** `Router(config)# crypto map mymap 10 ipsec-manual`	Specifies the crypto map entry to create or modify and enters crypto map configuration mode.
Step 4	**match address** *access-list-id* **Example:** `Router(config-crypto-m)# match address 102`	Names an IPsec access list that determines which traffic should be protected by IPsec and which traffic should not be protected by IPsec in the context of this crypto map entry. (The access list can specify only one **permit** entry when IKE is not used.)

	Command or Action	Purpose

Step 5	**set peer** {*hostname* | *ip-address*}
	Example:
	`Router(config-crypto-m)# set peer 10.0.0.5`
		Specifies the remote IPsec peer. This is the peer to which IPsec protected traffic should be forwarded.
		(Only one peer can be specified when IKE is not used.)

Step 6	**set transform-set** *transform-set-name* **Example:** `Router(config-crypto-m)# set transform-set someset`	Specifies which transform set should be used. This must be the same transform set that is specified in the remote peer’s corresponding crypto map entry. **Note** Only one transform set can be specified when IKE is not used.

Step 7	Enter the following commands:
	• **set session-key inbound ah** *spi hex-key-string* and
	• **set session-key outbound ah** *spi hex-key-string*
	Example:
	`Router(config-crypto-m)# set session-key inbound ah 256 98765432109876549876543210987654`
	Example:
	`Router(config-crypto-m)# set session-key outbound ah 256 fedcbafedcbafedcfedcbafedcbafedc`
		Sets the AH security parameter indexes (SPIs) and keys to apply to inbound and outbound protected traffic if the specified transform set includes the AH protocol.
		(This manually specifies the AH security association to be used with protected traffic.)

Step 8	Enter the following commands:
	• **set session-key inbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*] and
	• **set session-key outbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
	Example:
	`Router(config-crypto-m)# set session-key inbound esp 256 cipher 0123456789012345`
	Example:
	`Router(config-crypto-m)# set session-key outbound esp 256 cipher abcdefabcdefabcd`
		Sets the ESP SPIs and keys to apply to inbound and outbound protected traffic if the specified transform set includes the ESP protocol. Specifies the cipher keys if the transform set includes an ESP cipher algorithm. Specifies the authenticator keys if the transform set includes an ESP authenticator algorithm.
		(This manually specifies the ESP security association to be used with protected traffic.)

	Command or Action	Purpose

Step 9	**end**
	Example:
	`Router(config-crypto-m)# end`
		Exits crypto-map configuration mode and returns to privileged EXEC mode.

Step 10	**show crypto map interface** [*interface* | **tag** *map-name*] **Example:** `Router# show crypto map`	Displays your crypto map configuration.

- Troubleshooting Tips, page 26
- What to Do Next, page 26

Troubleshooting Tips

For manually established SAs, you must clear and reinitialize the SAs for the changes to take effect. To clear IPsec SAs, use the **clear crypto sa** command with appropriate parameters. (Omitting all parameters clears the full SA database, which clears active security sessions.)

What to Do Next

After you have successfully created a crypto map set, you must apply the crypto map set to each interface through which IPsec traffic flows. To complete this task, see the section “Applying Crypto Map Sets to Interfaces, page 26.”

Applying Crypto Map Sets to Interfaces

You need to apply a crypto map set to each interface through which IPsec traffic flows. Applying the crypto map set to an interface instructs the router to evaluate all the interface’s traffic against the crypto map set and to use the specified policy during connection or security association negotiation on behalf of traffic to be protected by crypto.

Perform this task to apply a crypto map to an interface.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **crypto map** *map-name*
5. **exit**
6. **crypto map** *map-name* **local-address** *interface-id*
7. **exit**
8. **show crypto map** [**interface** *interface*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. - Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type number* **Example:** `Router(config)# Interface FastEthernet 0/0`	Configures an interface and enters interface configuration mode.
Step 4	**crypto map** *map-name* **Example:** `Router(config-if)# crypto map mymap`	Applies a crypto map set to an interface.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**crypto map** *map-name* **local-address** *interface-id* **Example:** `Router(config)# crypto map mymap local-address loopback0`	(Optional) Permits redundant interfaces to share the same crypto map using the same local identity.
Step 7	**exit** **Example:** `Router(config)# exit`	(Optional) Exits global configuration mode.
Step 8	**show crypto map** [**interface** *interface*] **Example:** `Router# show crypto map`	(Optional) Displays your crypto map configuration

Configuration Examples for Configuring an IPsec VPN

Example: AES-Based Static Crypto Map

The following example is a portion of the **show running-config** command. This example shows how to configure a static crypto map and define AES as the encryption method.

```
crypto isakmp policy 10
 encryption aes 256
 authentication pre-share
 lifetime 180
crypto isakmp key cisco123 address 10.0.110.1
!
!
crypto ipsec transform-set aesset esp-aes 256 esp-sha-hmac
 mode transport
!
crypto map aesmap 10 ipsec-isakmp
 set peer 10.0.110.1
 set transform-set aesset
 match address 120
!
!
!
voice call carrier capacity active
!
!
mta receive maximum-recipients 0
!
!
interface FastEthernet0/0/1
 ip address 10.0.110.2 255.255.255.0
 ip nat outside
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
 crypto map aesmap
!
interface Serial0/0
 no ip address
 shutdown
!
interface FastEthernet0/1/1
 ip address 10.0.110.1 255.255.255.0
 ip nat inside
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
!
ip nat inside source list 110 interface FastEthernet0/0/1 overload
ip classless
ip route 0.0.0.0 0.0.0.0 10.5.1.1
ip route 10.0.110.0 255.255.255.0 FastEthernet0/0/1
ip route 172.18.124.0 255.255.255.0 10.5.1.1
ip route 172.18.125.3 255.255.255.255 10.5.1.1
ip http server
!
!
access-list 110 deny   ip 10.0.110.0 0.0.0.255 10.0.110.0 0.0.0.255
access-list 110 permit ip 10.0.110.0 0.0.0.255 any
access-list 120 permit ip 10.0.110.0 0.0.0.255 10.0.110.0 0.0.0.255
!
```

Additional References

Related Documents

Related Topic	Document Title
IKE configuration	Configuring Internet Key Exchange for IPsec VPNs
IKE, IPsec, and PKI configuration commands: complete command syntax, command mode, defaults, usage guidelines, and examples	• Cisco IOS Security Command Reference Commands A to C • Cisco IOS Security Command Reference Commands D to L • Cisco IOS Security Command Reference Commands M to R • Cisco IOS Security Command Reference Commands S to Z

Standards

Standards	Title
None	--

MIBs

MIBs	MIBs Link
• CISCO-IPSEC-FLOW-MONITOR- MIB • CISCO-IPSEC-MIB • CISCO-IPSEC-POLICY-MAP-MIB	To locate and download MIBs for selected platforms, Cisco IOS XE releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFCs	Title
RFC 2401	*Security Architecture for the Internet Protocol*
RFC 2402	*IP Authentication Header*
RFC 2403	*The Use of HMAC-MD5-96 within ESP and AH*
RFC 2404	*The Use of HMAC-SHA-1-96 within ESP and AH*
RFC 2405	*The ESP DES-CBC Cipher Algorithm With Explicit IV*

RFCs	Title
RFC 2406	*IP Encapsulating Security Payload (ESP)*
RFC 2407	*The Internet IP Security Domain of Interpretation for ISAKMP*
RFC 2408	*Internet Security Association and Key Management Protocol (ISAKMP)*

Technical Assistance

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

Feature Information for Security for VPNs with IPsec

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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

Table 3 Feature Information for Configuring Security for IPsec VPNs

Feature Name	Releases	Feature Information
Advanced Encryption Standard (AES)	Cisco IOS XE Release 2.1	This feature adds support for the new encryption standard AES, which is a privacy transform for IPsec and IKE and has been developed to replace DES. The following sections provide information about this feature: • Supported Standards, page 2 • Defining Transform Sets, page 16 The following commands were modified by this feature: **crypto ipsec transform-set, encryption (IKE policy), show crypto ipsec transform-set, show isakmp policy.**
IKE Shared Secret Using AAA Server	Cisco IOS XE Release 2.1	The IKE Shared Secret Using AAA Server feature enables key lookup from a AAA server. This feature was implemented on Cisco ASR 1000 Series Routers.

Glossary

anti-replay-- Security service where the receiver can reject old or duplicate packets to protect itself against replay attacks. IPsec provides this optional service by use of a sequence number combined with the use of data authentication. Cisco IOS XE IPsec provides this service whenever it provides the data authentication service, except for manually established SAs (that is, SAs established by configuration and not by IKE).

data authentication --Verification of the integrity and origin of the data. Data authentication can refer either to integrity alone or to both of these concepts (although data origin authentication is dependent upon data integrity).

data confidentiality -- Security service in which the protected data cannot be observed.

data flow -- Grouping of traffic, identified by a combination of source address or mask, destination address or mask, IP next protocol field, and source and destination ports, where the protocol and port fields can have the values of **any**. IPsec protection is applied to data flows.

peer -- In the context of this module, a “peer” is a router or other device that participates in IPsec.

PFS --perfect forward secrecy. Cryptographic characteristic associated with a derived shared secret value. With PFS, if one key is compromised, previous and subsequent keys are not compromised, because subsequent keys are not derived from previous keys.

SA --security association. Description of how two or more entities use security services in the context of a particular security protocol (AH or ESP) to communicate securely on behalf of a particular data flow. The transform and the shared secret keys are used for protecting the traffic.

SPI --security parameter index. A number which, together with a destination IP address and security protocol, uniquely identifies a particular security association. Without IKE, the SPI is manually specified for each security association.

transform -- List of operations performed on a dataflow to provide data authentication, data confidentiality, and data compression. For example, one transform is the ESP protocol with the HMAC-MD5 authentication algorithm; another transform is the AH protocol with the 56-bit DES encryption algorithm and the ESP protocol with the HMAC-SHA authentication algorithm.

tunnel -- In the context of this module, “tunnel” is a secure communication path between two peers, such as two routers. It does not refer to using IPsec in tunnel mode.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

IPsec Virtual Tunnel Interface

IP security (IPsec) virtual tunnel interfaces (VTIs) provide a routable interface type for terminating IPsec tunnels and an easy way to define protection between sites to form an overlay network. IPsec VTIs simplify configuration of IPsec for protection of remote links, support multicast, and simplify network management and load balancing.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

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

Restrictions for IPsec Virtual Tunnel Interface

IPsec Transform Set

The IPsec transform set must be configured in tunnel mode only.

IKE Security Association

The Internet Key Exchange (IKE) security association (SA) is bound to the VTI. Because IKE SA is bound to the VTI, the same IKE SA cannot be used for a crypto map.

IPsec SA Traffic Selectors

Static VTIs (SVTIs) support only a single IPsec SA that is attached to the VTI interface. The traffic selector for the IPsec SA is always “IP any any.”

IPv4 and IPv6 Packets

This feature supports SVTIs that are configured to encapsulate IPv4 packets or IPv6 packets, but IPv4 packets cannot carry IPv6 packets, and IPv6 packets cannot carry IPv4 packets.

Proxy

SVTIs support only the "IP any any" proxy.

QoS Traffic Shaping

The shaped traffic is process switched.

Stateful Failover

IPsec stateful failover is not supported with IPsec VTIs.

Tunnel Protection

The **shared** keyword is not required and must not be configured when using the **tunnel mode ipsec ipv4** command for IPsec IPv4 mode.

Static VTIs Versus GRE Tunnels

The IPsec VTI is limited to IP unicast and multicast traffic only, as opposed to generic routing encapsulation (GRE) tunnels, which have a wider application for IPsec implementation.

VRF-Aware IPsec Configuration

In VRF-aware IPsec configurations with either SVTIs or Dynamic VTIs (DVTIs), the VRF must not be configured in the Internet Security Association and Key Management Protocol (ISAKMP) profile. Instead, the VRF must be configured on the tunnel interface for SVTIs. For DVTIs, you must apply VRF to the virtual template using the **ip vrf forwarding** command.

Information About IPsec Virtual Tunnel Interface

The use of IPsec VTIs both greatly simplifies the configuration process when you need to provide protection for remote access and provides a simpler alternative to using a generic routing encapsulation (GRE) tunnel for encapsulation and crypto maps with IPsec. A major benefit associated with IPsec VTIs is that the configuration does not require a static mapping of IPsec sessions to a physical interface. The IPsec tunnel endpoint is associated with an actual (virtual) interface. Because there is a routable interface at the tunnel endpoint, many common interface capabilities can be applied to the IPsec tunnel.

The IPsec VTI allows for the flexibility of sending and receiving both IP unicast and multicast encrypted traffic on any physical interface, such as in the case of multiple paths. Traffic is encrypted or decrypted when it is forwarded from or to the tunnel interface and is managed by the IP routing table. Using IP routing to forward the traffic to the tunnel interface simplifies the IPsec VPN configuration compared to the more complex process of using access control lists (ACLs) with the crypto map in native IPsec configurations. DVTIs function like any other real interface so that you can apply quality of service (QoS), firewall, and other security services as soon as the tunnel is active.

Without Virtual Private Network (VPN) Acceleration Module2+ (VAM2+) accelerating virtual interfaces, the packet traversing an IPsec virtual interface is directed to the router processor (RP) for encapsulation. This method tends to be slow and has limited scalability. In hardware crypto mode, all the IPsec VTIs are accelerated by the VAM2+ crypto engine, and all traffic going through the tunnel is encrypted and decrypted by the VAM2+.

The following sections provide details about the IPsec VTI:

Benefits of Using IPsec Virtual Tunnel Interfaces

IPsec VTIs allow you to configure a virtual interface to which you can apply features. Features for clear-text packets are configured on the VTI. Features for encrypted packets are applied on the physical outside interface. When IPsec VTIs are used, you can separate the application of features such as NAT, ACLs, and QoS and apply them to clear-text or encrypted text, or both. When crypto maps are used, there is no simple way to apply encryption features to the IPsec tunnel.

There are two types of VTI interfaces: static VTIs (SVTIs) and dynamic VTIs (DVTIs).

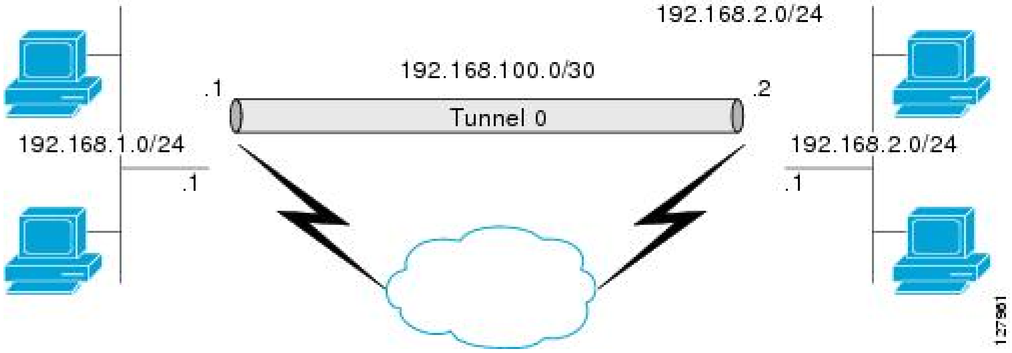
Static Virtual Tunnel Interfaces

SVTI configurations can be used for site-to-site connectivity in which a tunnel provides always-on access between two sites. The advantage of using SVTIs as opposed to crypto map configurations is that users can enable dynamic routing protocols on the tunnel interface without the extra 24 bytes required for GRE headers, thus reducing the bandwidth for sending encrypted data.

Additionally, multiple Cisco IOS XE software features can be configured directly on the tunnel interface and on the physical egress interface of the tunnel interface. This direct configuration allows users to have solid control on the application of the features in the pre- or post-encryption path.

The figure below illustrates how a SVTI is used.

Figure 4 IPsec SVTI



The IPsec VTI supports native IPsec tunneling and exhibits most of the properties of a physical interface.

Dynamic Virtual Tunnel Interfaces

DVTIs can provide highly secure and scalable connectivity for remote-access VPNs. The DVTI technology replaces dynamic crypto maps and the dynamic hub-and-spoke method for establishing tunnels. DVTI supports multiple IPsec SAs. The DVTI can accept multiple IPsec selectors that are proposed by the initiator.

DVTIs can be used for both the server and remote configuration. The tunnels provide an on-demand separate virtual access interface for each VPN session. The configuration of the virtual access interfaces is cloned from a virtual template configuration, which includes the IPsec configuration and any Cisco IOS XE software feature configured on the virtual template interface, such as QoS, NetFlow, or ACLs.

DVTIs function like any other real interface so that you can apply QoS, firewall, other security services as soon as the tunnel is active. QoS features can be used to improve the performance of various applications across the network. Any combination of QoS features offered in Cisco IOS XE software can be used to support voice, video, or data applications.

DVTIs provide efficiency in the use of IP addresses and provide secure connectivity. DVTIs allow dynamically downloadable per-group and per-user policies to be configured on a RADIUS server. The per-group or per-user definition can be created using extended authentication (Xauth) User or Unity group, or it can be derived from a certificate. DVTIs are standards based, so interoperability in a multiple-vendor environment is supported. IPsec DVTIs allow you to create highly secure connectivity for remote access VPNs and can be combined with Cisco Architecture for Voice, Video, and Integrated Data (AVVID) to deliver converged voice, video, and data over IP networks. The DVTI simplifies Virtual Private Network (VPN) routing and forwarding (VRF)-aware IPsec deployment. The VRF is configured on the interface.

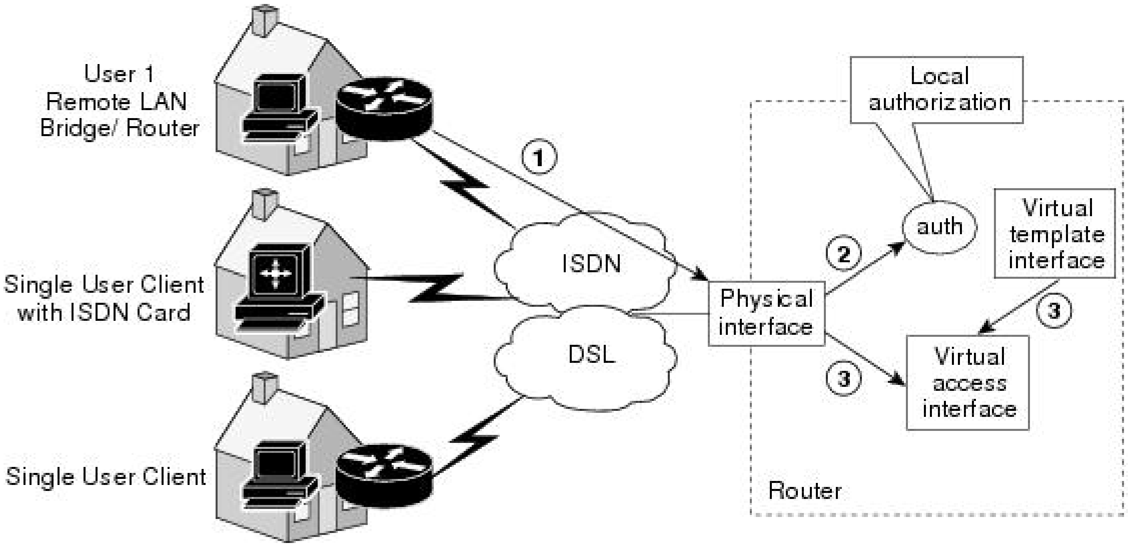
A DVTI requires minimal configuration on the router. A single virtual template can be configured and cloned.

The DVTI creates an interface for IPsec sessions and uses the virtual template infrastructure for dynamic instantiation and management of dynamic IPsec VTIs. The virtual template infrastructure is extended to create dynamic virtual-access tunnel interfaces. DVTIs are used in hub-and-spoke configurations. A single DVTI can support several static VTIs.

Note DVTI is supported in Easy VPNs. That is, the DVTI end must be configured as an Easy VPN server. DVTI can also be used in site-to-site scenarios.

The figure below illustrates the DVTI authentication path.

Figure 5 Dynamic IPsec VTI



The authentication shown in the figure above follows this path:

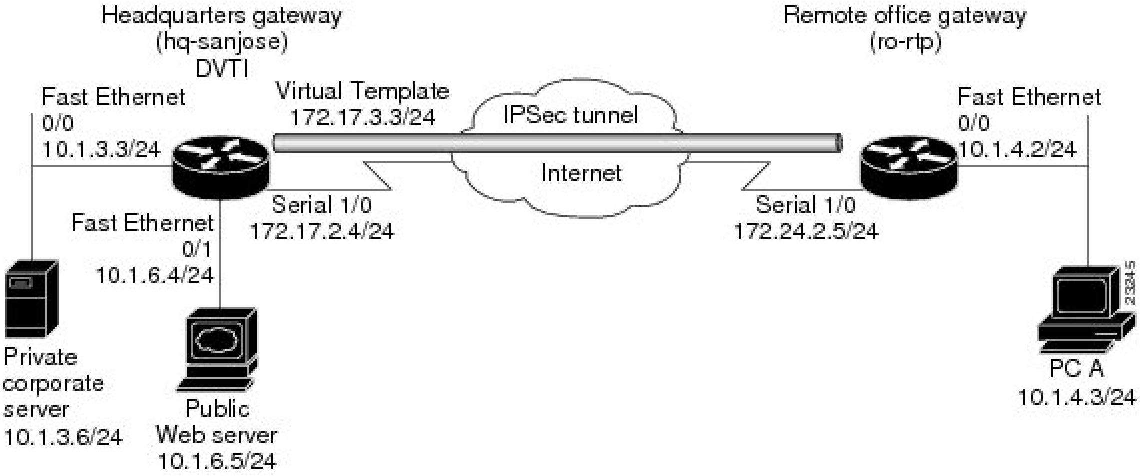
1 User 1 calls the router.

2 Router 1 authenticates User 1.

3 IPsec clones the virtual access interface from the virtual template interface.

The figure below illustrates the DVTI authentication path in a site-to-site scenario.

Figure 6 Dynamic IPsec VTI in a Site-to-Site Scenario



Dynamic Virtual Tunnel Interface Life Cycle

IPsec profiles define policy for DVTIs. The dynamic interface is created at the end of IKE Phase 1 and IKE Phase 1.5. The interface is deleted when the IPsec session to the peer is closed. The IPsec session is closed when both IKE and IPsec SAs to the peer are deleted.

Routing with IPsec Virtual Tunnel Interfaces

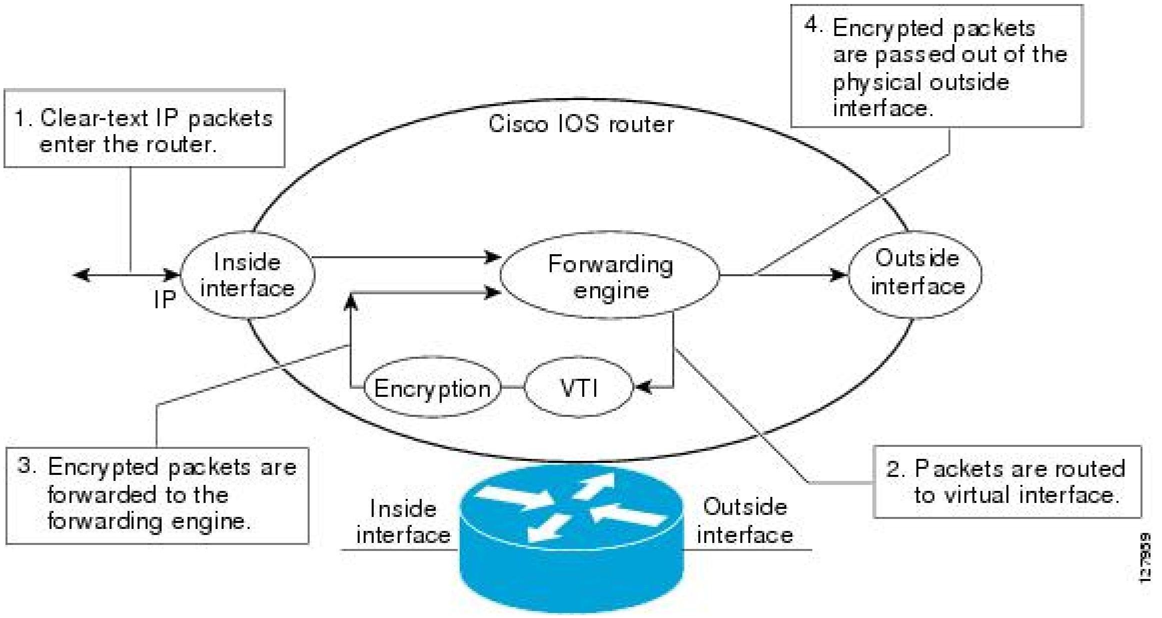
Because VTIs are routable interfaces, routing plays an important role in the encryption process. Traffic is encrypted only if it is forwarded out of the VTI, and traffic arriving on the VTI is decrypted and routed accordingly. VTIs allow you to establish an encryption tunnel using a real interface as the tunnel endpoint. You can route to the interface or apply services such as QoS, firewalls, network address translation, and NetFlow statistics as you would to any other interface. You can monitor the interface and route to it, and it has an advantage over crypto maps because it is a real interface and provides the benefits of any other Cisco IOS interface.

Traffic Encryption with the IPsec Virtual Tunnel Interface

When an IPsec VTI is configured, encryption occurs in the tunnel. Traffic is encrypted when it is forwarded to the tunnel interface. Traffic forwarding is handled by the IP routing table, and dynamic or static routing can be used to route traffic to the SVTI. DVTI uses reverse route injection to further simplify the routing configurations. Using IP routing to forward the traffic to encryption simplifies the IPsec VPN configuration because the use of ACLs with a crypto map in native IPsec configurations is not required. The IPsec virtual tunnel also allows you to encrypt multicast traffic with IPsec.

IPsec packet flow into the IPsec tunnel is illustrated in the figure below.

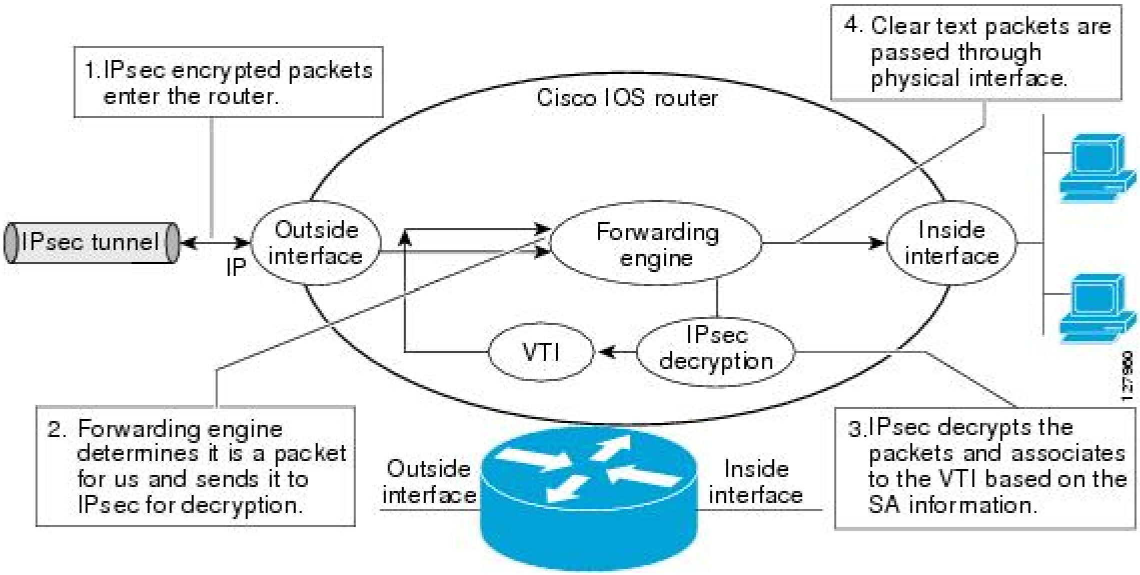
Figure 7 *Packet Flow into the IPsec Tunnel*


4. Encrypted packets are passed out of the physical outside interface.
1. Clear-text IP packets enter the router.
Cisco IOS router
IP
Inside interface
Forwarding engine
Outside interface
Encryption
VTI
3. Encrypted packets are forwarded to the forwarding engine.
Inside interface
Outside interface
2. Packets are routed to virtual interface.
127959

After packets arrive on the inside interface, the forwarding engine switches the packets to the VTI, where they are encrypted. The encrypted packets are handed back to the forwarding engine, where they are switched through the outside interface.

The figure below shows the packet flow out of the IPsec tunnel.

Figure 8 Packet Flow out of the IPsec Tunnel


1. IPsec encrypted packets enter the router.
Cisco IOS router
4. Clear text packets are passed through physical interface.
IPsec tunnel
IP
Outside interface
Forwarding engine
Inside interface
VTI
IPsec decryption
127990
2. Forwarding engine determines it is a packet for us and sends it to IPsec for decryption.
Outside interface
Inside interface
3. IPsec decrypts the packets and associates to the VTI based on the SA information.

How to Configure IPsec Virtual Tunnel Interface

- Configuring Static IPsec Virtual Tunnel Interfaces, page 39
- Configuring Dynamic IPsec Virtual Tunnel Interfaces, page 41

Configuring Static IPsec Virtual Tunnel Interfaces

Perform this task to configure a static IPsec VTI.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec profile** *profile-name*
4. **set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*]
5. **interface** *type number*
6. **ip address** *address mask*
7. **tunnel mode ipsec ipv4**
8. **tunnel source** *interface*
9. **tunnel destination** *ip-address*
10. **tunnel protection ipsec profile** *profile-name* [**shared**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec profile** *profile-name* **Example:** `Router(config)# crypto ipsec profile PROF`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec routers.
Step 4	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Router(config)# set transform-set tset`	Specifies which transform sets can be used with the crypto map entry.
Step 5	**interface** *type number* **Example:** `Router(config)# interface tunnel0`	Specifies the interface on which the tunnel is configured and enters interface configuration mode.
Step 6	**ip address** *address mask* **Example:** `Router(config-if)# ip address 10.1.1.1 255.255.255.0`	Specifies the IP address and mask.
Step 7	**tunnel mode ipsec ipv4** **Example:** `Router(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.

	Command or Action	Purpose
Step 8	**tunnel source** *interface* **Example:** `Router(config-if)# tunnel source loopback0`	Specifies the tunnel source as a loopback interface.
Step 9	**tunnel destination** *ip-address* **Example:** `Router(config-if)# tunnel destination 172.16.1.1`	Identifies the IP address of the tunnel destination.
Step 10	**tunnel protection ipsec profile** *profile-name* [**shared**] **Example:** `Router(config-if)# tunnel protection ipsec profile PROF`	Associates a tunnel interface with an IPsec profile.

Configuring Dynamic IPsec Virtual Tunnel Interfaces

Perform this task to configure a dynamic IPsec VTI.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec profile** *profile-name*
4. **set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*]
5. **interface virtual-template** *number*
6. **tunnel mode ipsec ipv4**
7. **tunnel protection ipsec profile** *profile-name* [**shared**]
8. **exit**
9. **crypto isakamp profile** *profile-name*
10. **virtua l- template** *template-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec profile** *profile-name* **Example:** `Router(config)# crypto ipsec profile PROF`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec routers.
Step 4	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Router(config)# set transform-set tset`	Specifies which transform sets can be used with the crypto map entry.
Step 5	**interface virtual-template** *number* **Example:** `Router(config)# interface virtual-template 2`	Defines a virtual-template tunnel interface and enters interface configuration mode.
Step 6	**tunnel mode ipsec ipv4** **Example:** `Router(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 7	**tunnel protection ipsec profile** *profile-name* [**shared**] **Example:** `Router(config-if)# tunnel protection ipsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 8	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode.

	Command or Action	Purpose
Step 9	**crypto isakmp profile** *profile-name* **Example:** `Router(config)# crypto isakmp profile red`	Defines the ISAKMP profile to be used for the virtual template.
Step 10	**virtua l- template** *template-number* **Example:** `Router(config)# virtual-template 1`	Specifies the virtual template attached to the ISAKMP profile.

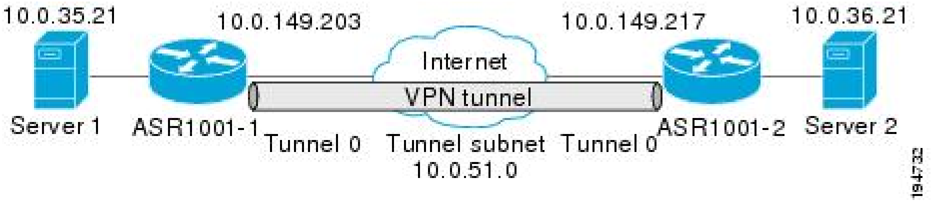
Configuration Examples for IPsec Virtual Tunnel Interface

- Example Static Virtual Tunnel Interface with IPsec, page 43
- Example VRF-Aware Static Virtual Tunnel Interface, page 46
- Example Static Virtual Tunnel Interface with QoS, page 46
- Example Static Virtual Tunnel Interface with Virtual Firewall, page 47
- Example Dynamic Virtual Tunnel Interface Easy VPN Server, page 48
- Example Dynamic Virtual Tunnel Interface Easy VPN Client, page 49
- Example VRF-Aware IPsec with Dynamic VTI When VRF Is Configured Under a Virtual Template, page 51
- Example VRF-Aware IPsec with Dynamic VTI When VRF Is Configured Under an ISAKMP Profile, page 51
- Example Dynamic VTI When VRF Is Configured Under a Virtual Template and an ISAKMP Profile, page 52
- Example Dynamic Virtual Tunnel Interface with a Virtual Firewall, page 53
- Example Dynamic Virtual Tunnel Interface with QoS, page 54

Example Static Virtual Tunnel Interface with IPsec

The following example configuration uses a preshared key for authentication between peers. VPN traffic is forwarded to the IPsec VTI for encryption and then sent out the physical interface. The tunnel on subnet 10 checks packets for IPsec policy and passes them to the Crypto Engine (CE) for IPsec encapsulation. The figure below illustrates the IPsec VTI configuration.

Figure 9 VTI with IPsec



Router ASR 1000-1 Configuration

```
version 2.1
service timestamps debug datetime
service timestamps log datetime
hostname ASR 1000-1
no aaa new-model
ip subnet-zero
ip cef
controller ISA 6/1
!
crypto isakmp policy 1
encr 3des
authentication pre-share
group 2
crypto isakmp key Cisco12345 address 0.0.0.0 0.0.0.0
crypto IPsec transform-set T1 esp-3des esp-sha-hmac
crypto IPsec profile P1
set transform-set T1
!
interface Tunnel0
 ip address 10.0.51.203 255.255.255.0
 ip ospf mtu-ignore
 load-interval 30
 tunnel source 10.0.149.203
 tunnel destination 10.0.149.217
 tunnel mode IPsec ipv4
 tunnel protection IPsec profile P1
!
interface FastEthernet3/0
 ip address 10.0.149.203 255.255.255.0
 duplex full
!
interface FastEthernet3/3
 ip address 10.0.35.203 255.255.255.0
 duplex full
!
ip classless
ip route 10.0.36.0 255.255.255.0 Tunnel0
line con 0
line aux 0
line vty 0 4
end
```

Router ASR 1000-2 Configuration

```
version 2.1
hostname ASR 1000-2
no aaa new-model
ip subnet-zero
ip cef
crypto isakmp policy 1
encr 3des
authentication pre-share
group 2
crypto isakmp key Cisco12345 address 0.0.0.0 0.0.0.0
crypto IPsec transform-set T1 esp-3des esp-sha-hmac
crypto IPsec profile P1
set transform-set T1
!
interface Tunnel0
 ip address 10.0.51.217 255.255.255.0
 ip ospf mtu-ignore
 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile P1
!
interface FastEthernet0/0
 ip address 10.0.149.217 255.255.255.0
 speed 100
 full-duplex
!
interface FastEthernet1/0
 ip address 10.0.36.217 255.255.255.0
 load-interval 30
 full-duplex
!
ip classless
ip route 10.0.35.0 255.255.255.0 Tunnel0
line con 0
line aux 0
line vty 0 4
end
```

- Example Verifying the Results for the IPsec Static Virtual Tunnel Interface, page 45

Example Verifying the Results for the IPsec Static Virtual Tunnel Interface

This section provides information that you can use to confirm that your configuration is working properly. In this display, Tunnel 0 is “up,” and the line protocol is “up.” If the line protocol is “down,” the session is not active.

Verifying the Router Status

```
Router# show interface tunnel 0
Tunnel0 is up, line protocol is up
Hardware is Tunnel
Internet address is 10.0.51.203/24
MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
reliability 255/255, txload 103/255, rxload 110/255
Encapsulation TUNNEL, loopback not set
Keepalive not set
Tunnel source 10.0.149.203, destination 10.0.149.217
Tunnel protocol/transport IPsec/IP
, key disabled, sequencing disabled
Tunnel TTL 255
Checksumming of packets disabled, fast tunneling enabled
Tunnel transmit bandwidth 8000 (kbps)
Tunnel receive bandwidth 8000 (kbps)
Tunnel protection via IPsec (profile "P1")
Last input never, output never, output hang never
Last clearing of "show interface" counters never
Input queue: 1/75/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/0 (size/max)
30 second input rate 13000 bits/sec, 34 packets/sec
30 second output rate 36000 bits/sec, 34 packets/sec
191320 packets input, 30129126 bytes, 0 no buffer
Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
59968 packets output, 15369696 bytes, 0 underruns
0 output errors, 0 collisions, 0 interface resets
0 output buffer failures, 0 output buffers swapped out
Router# show crypto session
Crypto session current status
Interface: Tunnel0
Session status: UP-ACTIVE
Peer: 10.0.149.217 port 500
IKE SA: local 10.0.149.203/500 remote 10.0.149.217/500 Active
IPsec FLOW: permit ip 0.0.0.0/0.0.0.0 0.0.0.0/0.0.0.0
Active SAs: 4, origin: crypto map
Router# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route
```

```
Gateway of last resort is not set
10.0.0.0/8 is variably subnetted, 4 subnets, 2 masks
C 10.0.35.0/24 is directly connected, FastEthernet3/3
S 10.0.36.0/24 is directly connected, Tunnel0
C 10.0.51.0/24 is directly connected, Tunnel0
C 10.0.149.0/24 is directly connected, FastEthernet3/0
```

Example VRF-Aware Static Virtual Tunnel Interface

To add VRF to the static VTI example, include the **ip vrf**and **ip vrf forwarding** commands to the configuration as shown in the following example:

Router Configuration

```
hostname ASR 1000-1
.
.
.
ip vrf sample-vti1
 rd 1:1
 route-target export 1:1
 route-target import 1:1
!
.
.
.
interface Tunnel0
 ip vrf forwarding sample-vti1
 ip address 10.0.51.217 255.255.255.0
 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile P1
.
.
.
!
end
```

Example Static Virtual Tunnel Interface with QoS

You can apply any QoS policy to the tunnel endpoint by including the **service-policy** statement under the tunnel interface. The following example is policing traffic out the tunnel interface:

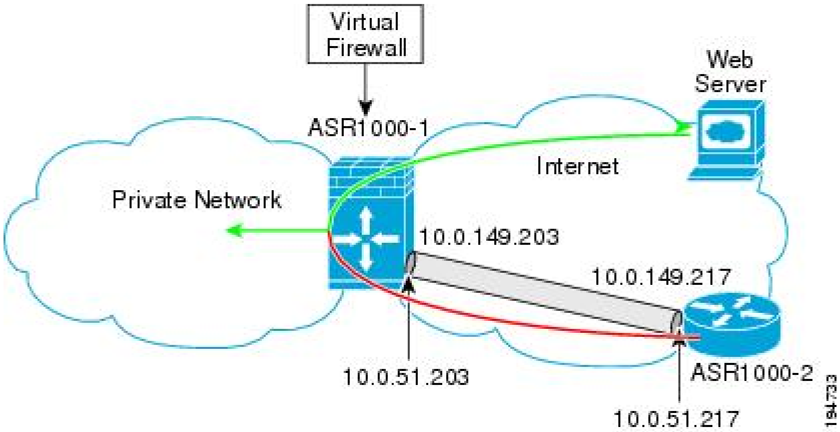
Router Configuration

```
hostname router1
.
.
.
class-map match-all VTI
 match any
!
policy-map VTI
  class VTI
  police cir 2000000
    conform-action transmit
    exceed-action drop
!
.
.
.
interface Tunnel0
 ip address 10.0.51.217 255.255.255.0
 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile P1
 service-policy output VTI
!
.
.
.
!
end
```

Example Static Virtual Tunnel Interface with Virtual Firewall

Applying the virtual firewall to the SVTI tunnel allows traffic from the spoke to pass through the hub to reach the Internet. The figure below illustrates a SVTI with the spoke protected inherently by the corporate firewall.

Figure 10 Static VTI with Virtual Firewall



The basic SVTI configuration has been modified to include the virtual firewall definition.

Router 1 Configuration

```
hostname ASR 1000-1
.
.
ip inspect max-incomplete high 1000000
ip inspect max-incomplete low 800000
ip inspect one-minute high 1000000
ip inspect one-minute low 800000
ip inspect tcp synwait-time 60
ip inspect tcp max-incomplete host 100000 block-time 2
ip inspect name IOSFW1 tcp timeout 300
ip inspect name IOSFW1 udp
!
.
.
.
interface GigabitEthernet0/1
 description Internet Connection
 ip address 172.18.143.246 255.255.255.0
 ip access-group 100 in
 ip nat outside
!
interface Tunnel0
 ip address 10.0.51.217 255.255.255.0
 ip nat inside
 ip inspect IOSFW1 in
 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile P1
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.18.143.1
!
ip nat translation timeout 120
ip nat translation finrst-timeout 2
ip nat translation max-entries 300000
ip nat pool test1 10.2.100.1 10.2.100.50 netmask 255.255.255.0
ip nat inside source list 110 pool test1 vrf test-vti1 overload
!
access-list 100 permit esp any any
access-list 100 permit udp any eq isakmp any
access-list 100 permit udp any eq non500-isakmp any
access-list 100 permit icmp any any
access-list 110 deny   esp any any
access-list 110 deny   udp any eq isakmp any
access-list 110 permit ip any any
access-list 110 deny   udp any eq non500-isakmp any
!
end
```

Example Dynamic Virtual Tunnel Interface Easy VPN Server

The following example illustrates the use of the DVTI Easy VPN server, which serves as an IPsec remote access aggregator. The client can be a home user running a Cisco VPN client or it can be a Cisco IOS XE router configured as an Easy VPN client.

Router Configuration

```
hostname ASR 1000-1
!
aaa new-model
aaa authentication login local_list local
aaa authorization network local_list local
aaa session-id common
!
ip subnet-zero
ip cef
!
username cisco password 0 cisco123
!
controller ISA 1/1
!
crypto isakmp policy 1
 encr 3des
 authentication pre-share
 group 2
!
crypto isakmp client configuration group group1
 key cisco123
 pool group1pool
 save-password
!
crypto isakmp profile vpn1-ra
   match identity group group1
   client authentication list local_list
   isakmp authorization list local_list
   client configuration address respond
   virtual-template 1
!
crypto ipsec transform-set VTI-TS esp-3des esp-sha-hmac
!
crypto ipsec profile test-vti1
 set transform-set VTI-TS
```

```
!
interface GigabitEthernet0/1
 description Internet Connection
 ip address 172.18.143.246 255.255.255.0
!
interface GigabitEthernet0/2
 description Internal Network
 ip address 10.2.1.1 255.255.255.0
!
interface Virtual-Template1 type tunnel
 ip unnumbered GigabitEthernet0/1
 ip virtual-reassembly
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
!
ip local pool group1pool 192.168.1.1 192.168.1.4
ip classless
ip route 0.0.0.0 0.0.0.0 172.18.143.1
!
end
```

- Example Verifying the Results for the Dynamic Virtual Tunnel Interface Easy VPN Server, page 49

Example Verifying the Results for the Dynamic Virtual Tunnel Interface Easy VPN Server

The following examples show that a DVTI has been configured for an Easy VPN server:

```
Router# show running-config interface Virtual-Access2
Building configuration...
Current configuration : 250 bytes
!
interface Virtual-Access2
 ip unnumbered GigabitEthernet0/1
 ip virtual-reassembly
 tunnel source 172.18.143.246
 tunnel destination 172.18.143.208
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
 no tunnel protection ipsec initiate
end
Router# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 10.2.1.10 to network 0.0.0.0
     172.18.0.0/24 is subnetted, 1 subnets
C       172.18.143.0 is directly connected, GigabitEthernet0/1
     192.168.1.0/32 is subnetted, 1 subnets
S       192.168.1.1 [1/0] via 0.0.0.0, Virtual-Access2
     10.0.0.0/24 is subnetted, 1 subnets
C       10.2.1.0 is directly connected, GigabitEthernet0/2
S*   0.0.0.0/0 [1/0] via 172.18.143.1
```

Example Dynamic Virtual Tunnel Interface Easy VPN Client

The following example shows how you can set up a router as the Easy VPN client. This example uses basically the same idea as the Easy VPN client that you can run from a PC to connect. In fact, the configuration of the Easy VPN server works for the software client or the Cisco IOS XE client.

```
hostname ASR 1000
!
no aaa new-model
```

```
!
ip cef
!
username cisco password 0 cisco123
!
crypto ipsec client ezvpn CLIENT
 connect manual
 group group1 key cisco123
 mode client
 peer 172.18.143.246
 virtual-interface 1
 username cisco password cisco123
 xauth userid mode local
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface FastEthernet0/0
 description Internet Connection
 ip address 172.18.143.208 255.255.255.0
 crypto ipsec client ezvpn CLIENT
!
interface FastEthernet0/1
 ip address 10.1.1.252 255.255.255.0
 crypto ipsec client ezvpn CLIENT inside
!
interface Virtual-Template1 type tunnel
 ip unnumbered Loopback0
!
ip route 0.0.0.0 0.0.0.0 172.18.143.1 254
!
end
```

The client definition can be set up in many different ways. The mode specified with the **connect** command can be automatic or manual. If the connect mode is set to manual, then the IPsec tunnel has to be initiated manually by a user.

Also note use of the **mode** command. The mode can be client, network-extension, or network-extension-plus. This example indicates client mode, which means that the client is given a private address from the server. Network-extension mode is different from client mode in that the client specifies for the server its attached private subnet. Depending on the mode, the routing table on either end is slightly different. The basic operation of the IPSec tunnel remains the same, regardless of the specified mode.

- Example Verifying the Results for the Dynamic Virtual Tunnel Interface Easy VPN Client, page 50

Example Verifying the Results for the Dynamic Virtual Tunnel Interface Easy VPN Client

The following examples illustrate different ways to display the status of the DVTI:

```
Router# show running-config interface Virtual-Access2
Building configuration...
Current configuration : 148 bytes
!
interface Virtual-Access2
 ip unnumbered Loopback1
 tunnel source FastEthernet0/0
 tunnel destination 172.18.143.246
 tunnel mode ipsec ipv4
end
Router# show running-config interface Loopback1
Building configuration...
Current configuration : 65 bytes
!
interface Loopback1
 ip address 192.168.1.1 255.255.255.255
end
Router# show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
```

```
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route
Gateway of last resort is 172.18.143.1 to network 0.0.0.0
     10.0.0.0/32 is subnetted, 1 subnets
C       10.1.1.1 is directly connected, Loopback0
     172.18.0.0/24 is subnetted, 1 subnets
C       172.18.143.0 is directly connected, FastEthernet0/0
     192.168.1.0/32 is subnetted, 1 subnets
C       192.168.1.1 is directly connected, Loopback1
S*   0.0.0.0/0 [1/0] via 0.0.0.0, Virtual-Access2
Router# show crypto ipsec client ezvpn
Easy VPN Remote Phase: 6
Tunnel name : CLIENT
Inside interface list: FastEthernet0/1
Outside interface: Virtual-Access2 (bound to FastEthernet0/0)
Current State: IPSEC_ACTIVE
Last Event: SOCKET_UP
Address: 192.168.1.1
Mask: 255.255.255.255
Save Password: Allowed
Current EzVPN Peer: 172.18.143.246
```

Example VRF-Aware IPsec with Dynamic VTI When VRF Is Configured Under a Virtual Template

The following example shows how to configure VRF-Aware IPsec to take advantage of the DVTI:

```
hostname ASR 1000
.
.
.
ip vrf test-vti1
 rd 1:1
 route-target export 1:1
 route-target import 1:1
!
.
.
.
interface Virtual-Template1 type tunnel
 ip vrf forwarding test-vti1
 ip unnumbered Loopback0
 ip virtual-reassembly
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
!
.
.
.
end
```

Example VRF-Aware IPsec with Dynamic VTI When VRF Is Configured Under an ISAKMP Profile

The following example shows how to configure VRF-Aware IPsec to take advantage of the DVTI when VRF is configured under an ISAKMP profile:

```
hostname ASR 1000
.
.
.
```

```
ip vrf test-vti1
rd 1:1
route-target export 1:1
route-target import 1:1
!
.
.
.
crypto isakmp profile cisco-isakmp-profile
   vrf test-vti1
   keyring key
   match identity address 4.0.0.22 255.255.255.255
!
.
.
.
interface Virtual-Template1 type tunnel
ip unnumbered Loopback0
ip virtual-reassembly
tunnel mode ipsec ipv4
tunnel protection ipsec profile test-vti1
!
.
.
end
```

Example Dynamic VTI When VRF Is Configured Under a Virtual Template and an ISAKMP Profile

Note When separate VRFs are configured under isakmp profile and virtual-template, the VRF configured under virtual template takes precedence.

The following example shows how to configure VRF-Aware IPsec to take advantage of the DVTI when VRF is configured under both a virtual-template and an ISAKMP profile:

```
hostname ASR 1000
.
.
.
ip vrf test-vti2
rd 1:2
route-target export 1:1
route-target import 1:1
!
.
.
.
ip vrf test-vti1
rd 1:1
route-target export 1:1
route-target import 1:1
!
.
.
.
crypto isakmp profile cisco-isakmp-profile
   vrf test-vti2
   keyring key
   match identity address 4.0.0.22 255.255.255.255
!
.
.
.
interface Virtual-Template1 type tunnel
ip vrf forwarding test-vti1
ip unnumbered Loopback0
ip virtual-reassembly
tunnel mode ipsec ipv4
tunnel protection ipsec profile test-vti1
!
.
.
.
end
```

Example Dynamic Virtual Tunnel Interface with a Virtual Firewall

The DVTI Easy VPN server can be configured behind a virtual firewall. Behind-the-firewall configuration allows users to enter the network, while the network firewall is protected from unauthorized access. The virtual firewall uses Context-Based Access Control (CBAC) and NAT applied to the Internet interface as well as to the virtual template.

```
hostname ASR 1000
.
.
.
ip inspect max-incomplete high 1000000
ip inspect max-incomplete low 800000
ip inspect one-minute high 1000000
ip inspect one-minute low 800000
ip inspect tcp synwait-time 60
ip inspect tcp max-incomplete host 100000 block-time 2
ip inspect name IOSFW1 tcp timeout 300
ip inspect name IOSFW1 udp
!
.
.
.
interface GigabitEthernet0/1
 description Internet Connection
 ip address 172.18.143.246 255.255.255.0
 ip access-group 100 in
 ip nat outside
!
interface GigabitEthernet0/2
 description Internal Network
 ip address 10.2.1.1 255.255.255.0
!
interface Virtual-Template1 type tunnel
 ip unnumbered Loopback0
 ip nat inside
 ip inspect IOSFW1 in
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.18.143.1
!
ip nat translation timeout 120
ip nat translation finrst-timeout 2
ip nat translation max-entries 300000
ip nat pool test1 10.2.100.1 10.2.100.50 netmask 255.255.255.0
ip nat inside source list 110 pool test1 vrf test-vti1 overload
!
access-list 100 permit esp any any
access-list 100 permit udp any eq isakmp any
access-list 100 permit udp any eq non500-isakmp any
access-list 100 permit icmp any any
access-list 110 deny   esp any any
access-list 110 deny   udp any eq isakmp any
access-list 110 permit ip any any
access-list 110 deny   udp any eq non500-isakmp any
!
end
```

Example Dynamic Virtual Tunnel Interface with QoS

You can add QoS to the DVTI tunnel by applying the service policy to the virtual template. When the template is cloned to make the virtual-access interface, the service policy is applied there. The following example shows the basic DVTI configuration with QoS added:

```
hostname ASR 1000
.
.
.
class-map match-all VTI
 match any
!
policy-map VTI
  class VTI
  police cir 2000000
    conform-action transmit
    exceed-action drop
!
.
.
.
interface Virtual-Template1 type tunnel
 ip vrf forwarding test-vti1
 ip unnumbered Loopback0
 ip virtual-reassembly
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
 service-policy output VTI
!
.
.
.
!
end
```

Additional References

Related Documents

Related Topic	Document Title
IPsec security issues	“ Configuring Security for VPNs with IPsec” module in the *Cisco IOS XE Security Configuration Guide: Secure Connectivity*
QoS configuration tasks	*Cisco IOS XE Quality of Service Solutions Configuration Guide*
Security commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*
VPN configuration tasks	“ Easy VPN Server” module in the *Cisco IOS XE Security Configuration Guide: Secure Connectivity*
Cisco IOS commands	Cisco IOS Master Commands List, All Releases

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2401	*Security Architecture for the Internet Protocol*
RFC 2408	*Internet Security Association and Key Management Protocol*
RFC 2409	*The Internet Key Exchange (IKE)*

Technical Assistance

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

Feature Information for IPsec Virtual Tunnel Interface

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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

Table 4 Feature Information for IPsec Virtual Tunnel Interface

Feature Name	Releases	Feature Configuration Information
Static IPsec VTIs	Cisco IOS XE Release 2.1	IPsec VTIs provide a routable interface type for terminating IPsec tunnels and an easy way to define protection between sites to form an overlay network. IPsec VTIs simplify configuration of IPsec for protection of remote links, support multicast, and simplify network management and load balancing. • Static Virtual Tunnel Interfaces, page 35 • Configuring Static IPsec Virtual Tunnel Interfaces, page 39 The following commands were introduced or modified: **crypto isakmp profile, interface virtual-template, show vtemplate, tunnel mode**.

Feature Name	Releases	Feature Configuration Information
Dynamic IPsec VTIs	Cisco IOS XE Release 2.1	Dynamic VTIs provide efficiency in the use of IP addresses and provide secure connectivity. Dynamic VTIs allow dynamically downloadable per-group and per-user policies to be configured on a RADIUS server. The per-group or per-user definition can be created using Xauth User or Unity group, or it can be derived from a certificate. Dynamic VTIs are standards based, so interoperability in a multiple-vendor environment is supported. IPsec dynamic VTIs allow you to create highly secure connectivity for remote access VPNs and can be combined with Cisco AVVID to deliver converged voice, video, and data over IP networks. The dynamic VTI simplifies VRF-aware IPsec deployment. The VRF is configured on the interface. The following sections provide information about this feature: • Dynamic Virtual Tunnel Interfaces, page 36 • Configuring Dynamic IPsec Virtual Tunnel Interfaces, page 41 The following command was introduced or modified: **virtual-template**. Cisco IOS XE Release 3.2S -- DVTI supports multiple IPsec SAs. The DVTI can accept multiple IPsec selectors that are proposed by the initiator. The following commands were introduced or modified: **set security-policy limit, set reverse-route.**

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Crypto Conditional Debug Support

The Crypto Conditional Debug Support feature introduces three new command-line interfaces (CLIs) that allow users to debug an IP Security (IPSec) tunnel on the basis of predefined crypto conditions such as the peer IP address, connection-ID of a crypto engine, and security parameter index (SPI). By limiting debug messages to specific IPSec operations and reducing the amount of debug output, users can better troubleshoot a router with a large number of tunnels.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

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

Prerequisites for Crypto Conditional Debug Support

To use the new crypto CLIs, you must be using a crypto image, such as the k8 or k9 subsystem.

Restrictions for Crypto Conditional Debug Support

- This feature does not support debug message filtering for hardware crypto engines.
- Although conditional debugging is useful for troubleshooting peer-specific or functionality related Internet Key Exchange (IKE) and IPSec problems, conditional debugging may not be able to define and check large numbers of debug conditions. Because extra space is needed to store the debug condition values, additional processing overhead is added to the CPU and memory usage is increased.

Thus, enabling crypto conditional debugging on a router with heavy traffic should be used with caution.

Information About Crypto Conditional Debug Support

Supported Condition Types

The new crypto conditional debug CLIs--**debug crypto condition**, **debug crypto condition unmatched**, **and show crypto debug-condition**--allow you to specify conditions (filter values) in which to generate and display debug messages related only to the specified conditions. The table below lists the supported condition types.

Table 5 Supported Condition Types for Crypto Debug CLI

Condition Type (Keyword)	Description
connid [1]	An integer between 1-32766. Relevant debug messages will be shown if the current IPSec operation uses this value as the connection ID to interface with the crypto engine.
flowid 1	An integer between 1-32766. Relevant debug messages will be shown if the current IPSec operation uses this value as the flow-ID to interface with the crypto engine.
FVRF	The name string of a virtual private network (VPN) routing and forwarding (VRF) instance. Relevant debug messages will be shown if the current IPSec operation uses this VRF instance as its front-door VRF (FVRF).
IVRF	The name string of a VRF instance. Relevant debug messages will be shown if the current IPSec operation uses this VRF instance as its inside VRF (IVRF).
peer group	A Unity group-name string. Relevant debug messages will be shown if the peer is using this group name as its identity.

[1] If an IPSec connid, flowid, or SPI is used as a debug condition, the debug messages for a related IPSec flow are generated. An IPSec flow has two connids, flowids, and SPIs--one inbound and one outbound. Both two connids, flowids, and SPIs can be used as the debug condition that triggers debug messages for the IPSec flow.

Condition Type (Keyword)	Description
peer hostname	A fully qualified domain name (FQDN) string. Relevant debug messages will be shown if the peer is using this string as its identity; for example, if the peer is enabling IKE Xauth with this FQDN string.
peer ipaddress	A single IP address. Relevant debug messages will be shown if the current IPSec operation is related to the IP address of this peer.
peer subnet	A subnet and a subnet mask that specify a range of peer IP addresses. Relevant debug messages will be shown if the IP address of the current IPSec peer falls into the specified subnet range.
peer username	A username string. Relevant debug messages will be shown if the peer is using this username as its identity; for example, if the peer is enabling IKE Extended Authentication (Xauth) with this username.
SPI 1	A 32-bit unsigned integer. Relevant debug messages will be shown if the current IPSec operation uses this value as the SPI.

How to Enable Crypto Conditional Debug Support

- Enabling Crypto Conditional Debug Messages, page 61
- Enabling Crypto Error Debug Messages, page 63

Enabling Crypto Conditional Debug Messages

- Performance Considerations, page 61
- Disable Crypto Debug Conditions, page 62

Performance Considerations

- Before enabling crypto conditional debugging, you must decide what debug condition types (also known as debug filters) and values will be used. The volume of debug messages is dependent on the number of conditions you define.

Note Specifying numerous debug conditions may consume CPU cycles and negatively affect router performance.

- Your router will perform conditional debugging only after at least one of the global crypto debug commands--**debug crypto isakmp**, **debug crypto ipsec**, and **debug crypto engine**--has been enabled.

This requirement helps to ensure that the performance of the router will not be impacted when conditional debugging is not being used.

Disable Crypto Debug Conditions

If you choose to disable crypto conditional debugging, you must first disable any crypto global debug CLIs you have issued ; thereafter, you can disable conditional debugging.

Note The **reset** keyword can be used to disable all configured conditions at one time.

SUMMARY STEPS

1. **enable**
2. **debug crypto condition** [**connid** *integer* **engine-id** *integer*] [**flowid** *integer***engine-id** *integer*] [**fvrf** *string*] [**ivrf** *string*] [**peer** [**group** *string*] [**hostname** *string*] [**ipv4** *ipaddress*] [**subnet** *subnet mask*] [**username** *string*]] [**spi** *integer*] [**reset**]
3. **show crypto debug-condition** {[**peer**] [**connid**] [**spi**] [**fvrf**] [**ivrf**] [**unmatched**]}
4. **debug crypto isakmp**
5. **debug crypto ipsec**
6. **debug crypto engine**
7. **debug crypto condition unmatched** [**isakmp** | **ipsec** | **engine**]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug crypto condition** [**connid** *integer* **engine-id** *integer*] [**flowid** *integer***engine-id** *integer*] [**fvrf** *string*] [**ivrf** *string*] [**peer** [**group** *string*] [**hostname** *string*] [**ipv4** *ipaddress*] [**subnet** *subnet mask*] [**username** *string*]] [**spi** *integer*] [**reset**] **Example:** `Router# debug crypto condition connid 2000 engine-id 1`	Defines conditional debug filters.

	Command or Action	Purpose

Step 3	**show crypto debug-condition** {[**peer**] [**connid**] [**spi**] [**fvrf**] [**ivrf**] [**unmatched**]}
	Example:
	`Router# show crypto debug-condition spi`
		Displays crypto debug conditions that have already been enabled in the router.

Step 4	**debug crypto isakmp**
	Example:
	`Router#`
	`debug crypto isakmp`
		Enables global IKE debugging.

Step 5	**debug crypto ipsec**
	Example:
	`Router#`
	`debug crypto ipsec`
		Enables global IPSec debugging.

Step 6	**debug crypto engine**
	Example:
	`Router#`
	`debug crypto engine`
		Enables global crypto engine debugging.

Step 7	**debug crypto condition unmatched** [**isakmp** | **ipsec** | **engine**] **Example:** `Router# debug crypto condition unmatched ipsec`	(Optional) Displays debug conditional crypto messages when no context information is available to check against debug conditions. If none of the optional keywords are specified, all crypto-related information will be shown.

Enabling Crypto Error Debug Messages

To enable crypto error debug messages, you must perform the following tasks.

- debug crypto error CLI, page 63

debug crypto error CLI

Enabling the **debug crypto error** command displays only error-related debug messages, thereby, allowing you to easily determine why a crypto operation, such as an IKE negotiation, has failed within your system.

Note

When enabling this command, ensure that global crypto debug commands are not enabled; otherwise, the global commands will override any possible error-related debug messages.

SUMMARY STEPS

1. **enable**
2. **debug crypto isakmp** | **ipsec** | **engine**} **error**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**debug crypto isakmp** | **ipsec** | **engine**} **error** **Example:** `Router# debug crypto ipsec error`	Enables only error debugging messages for a crypto area.

Configuration Examples for the Crypto Conditional Debug CLIs

Enabling Crypto Conditional Debugging Example

The following example shows how to display debug messages when the peer IP address is 10.1.1.1, 10.1.1.2, or 10.1.1.3, and when the connection-ID 2000 of crypto engine 0 is used. This example also shows how to enable global debug crypto CLIs and enable the **show crypto debug-condition** command to verify conditional settings.

```
Router#
debug crypto condition connid 2000 engine-id 1
Router#
debug crypto condition peer ipv4 10.1.1.1
Router#
debug crypto condition peer ipv4 10.1.1.2
Router#
debug crypto condition peer ipv4 10.1.1.3
Router#
debug crypto condition unmatched
! Verify crypto conditional settings.
Router#
show crypto debug-condition
```

```
Crypto conditional debug currently is turned ON
IKE debug context unmatched flag:ON
IPsec debug context unmatched flag:ON
Crypto Engine debug context unmatched flag:ON
IKE peer IP address filters:
10.1.1.1  10.1.1.2   10.1.1.3
Connection-id filters:[connid:engine_id]2000:1,
! Enable global crypto CLIs to start conditional debugging.
Router#
debug crypto isakmp
Router#
debug crypto ipsec
Router#
debug crypto engine
```

Disabling Crypto Conditional Debugging Example

The following example shows how to disable all crypto conditional settings and verify that those settings have been disabled:

```
Router#
debug crypto condition reset
! Verify that all crypto conditional settings have been disabled.
Router#
show crypto debug-condition
Crypto conditional debug currently is turned OFF
IKE debug context unmatched flag:OFF
IPsec debug context unmatched flag:OFF
Crypto Engine debug context unmatched flag:OFF
```

Additional References

The following sections provide references to the Crypto Conditional Debug Support feature.

Related Documents

Related Topic	Document Title
IPSec and IKE configuration tasks	“ Internet Key Exchange for IPsec VPNs “ module in the *Cisco IOS XE Security Configuration Guide: Secure Connectivity*
IPSec and IKE commands	*Cisco IOS Security Command Reference*

Standards

Standards	Title
None	--

MIBs

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

RFCs

RFCs	Title
None	--

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for Crypto Conditional Debug Support

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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

Table 6 Feature Information for Crypto Conditional Debug Support

Feature Name	Releases	Feature Information
Crypto Conditional Debug Support	Cisco IOS XE Release 2.1	The Crypto Conditional Debug Support feature introduces three new command-line interfaces (CLIs) that allow users to debug an IP Security (IPSec) tunnel on the basis of predefined crypto conditions such as the peer IP address, connection-ID of a crypto engine, and security parameter index (SPI).

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.