Security for VPNs with IPsec Configuration Guide Cisco IOS Release 15SY

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 IPsec VPNs. IPsec is a framework of open standards developed by the IETF. It provides security for the 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.

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

Finding Feature Information

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

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 for IPsec VPNs*.

Note If you decide not to use IKE, you must still 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, respectively. Ensure that your access lists are configured so that traffic from protocol 50, 51, and UDP port 500 are 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

NAT Configuration

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

Nested IPsec Tunnels

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 supported on releases up to and including Cisco IOS Release 12.4(15)T, but not on later releases.

With CSCts46591, the following nested IPsec tunnels are supported:

- Virtual tunnel interface (VTI) in VTI
- VTI in Generic Routing Encapsulation (GRE)/IPsec
- GRE/IPsec in VTI
- GRE/IPsec in GRE/IPsec

Unicast IP Datagram Application Only

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.

Information About Configuring Security for VPNs with IPsec

- Cisco IOS Suite-B Support for IKE and IPsec Cryptographic Algorithms, page 16
- Crypto Map Sets, page 18

Supported Standards

Cisco implements the following standards with this feature:

- IPsec—IPsec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPsec provides these security services at the IP layer; IPsec uses IKE to handle negotiation of protocols and algorithms based on the local policy, and 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.

  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.

- IKE—A hybrid protocol that implements Oakley and SKEME key exchanges inside the Internet Security Association and Key Management Protocol (ISAKMP) framework. While IKE is used with other protocols, its initial implementation is with the IPsec protocol. IKE provides authentication of IPsec peers, negotiates IPsec security associations, and establishes IPsec keys.

The component technologies implemented for IPsec include:

Note Cisco no longer recommends using DES, 3DES, MD5 (including HMAC variant), and Diffie-Hellman (DH) groups 1, 2 and 5; instead, you should use AES, SHA and DH Groups 14 or higher. For more information about the latest Cisco cryptographic recommendations, see the *Next Generation Encryption* (NGE) white paper.

- AES—Advanced Encryption Standard. A cryptographic algorithm that protects sensitive, unclassified information. AES is a privacy transform for IPsec and IKE and has been developed to replace 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 software 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 IPsec also implements the RFC 1829 version of ESP DES-CBC.

Cisco IOS also implements Triple DES (168-bit) encryption, depending on the software versions available for a specific platform. Cisco no longer recommends Triple DES (3DES).

Note Cisco IOS 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 send an e-mail to export@cisco.com.

- SEAL—Software Encryption Algorithm. An alternative algorithm to software-based DES, 3DES, and AES. SEAL encryption uses a 160-bit encryption key and has a lower impact on the CPU when compared to other software-based algorithms.
- SHA-2 and SHA-1 family (HMAC variant)—Secure Hash Algorithm (SHA) 1 and 2. Both SHA-1 and SHA-2 are hash algorithms used to authenticate packet data and verify the integrity verification mechanisms for the IKE protocol. HMAC is a variant that provides an additional level of hashing. SHA-2 family adds the SHA-256 bit hash algorithm and SHA-384 bit hash algorithm. This functionality is part of the Suite-B requirements that comprises four user interface suites of cryptographic algorithms for use with IKE and IPSec that are described in RFC 4869. Each suite consists of an encryption algorithm, a digital signature algorithm, a key agreement algorithm, and a hash or message digest algorithm. See the Configuring Security for VPNs with IPsec feature module for more detailed information about Cisco IOS Suite-B support.
- Diffie-Hellman—A public-key cryptography protocol that allows two parties to establish a shared secret over an unsecure communications channel. Diffie-Hellman is used within IKE to establish session keys. It supports 768-bit (the default), 1024-bit, 1536-bit, 2048-bit, 3072-bit, and 4096-bit DH groups. It also supports a 2048-bit DH group with a 256-bit subgroup, and 256-bit and 384-bit elliptic curve DH (ECDH). Cisco recommends using 2048-bit or larger DH key exchange, or ECDH key exchange.
- MD5 (Hash-based Message Authentication Code (HMAC) variant)—Message digest algorithm 5 (MD5) is a hash algorithm. HMAC is a keyed hash variant used to authenticate data.

IPsec as implemented in Cisco 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 Hardware

VPN Accelerator Module Support

The VPN Accelerator Module (VAM) is a single-width acceleration module. It provides high-performance, hardware-assisted tunneling and encryption services suitable for VPN remote access, site-to-site intranet,

and extranet applications. It also provides platform scalability and security while working with all services necessary for successful VPN deployments—security, quality of service (QoS), firewall and intrusion detection, service-level validation, and management. The VAM off-loads IPsec processing from the main processor, thus freeing resources on the processor engines for other tasks.

The VAM provides hardware-accelerated support for the following multiple encryption functions:

- 56-bit DES standard mode: CBC
- 3-Key Triple DES (168-bit)
- SHA-1 and MD5
- Rivest, Shamir, Adleman (RSA) public-key algorithm
- Diffie-Hellman key exchange RC4-40

For more information on VAMs, see the document *VPN Acceleration Module (VAM)*.

AIMs and NM Support

The data encryption Advanced Integration Module (AIM) and Network Module (NM) provide hardware-based encryption.

The data encryption AIMs and NM are hardware Layer 3 (IPsec) encryption modules and provide DES and Triple DES IPsec encryption for multiple T1s or E1s of bandwidth. These products also have hardware support for Diffie-Hellman, RSA, and DSA key generation.

Before using either module, note that RSA manual keying is not supported.

See the table below to determine which VPN encryption module to use.

IPPCP Software for Use with AIMs and NMs in Cisco 2600 and Cisco 3600 Series Routers

The software Internet Protocol Payload Compression Protocol (IPPCP) with AIMs and NMs allows customers to use Lempel-Ziv-Stac (LZS) software compression with IPsec when a VPN module is in Cisco 2600 and Cisco 3600 series routers, allowing users to effectively increase the bandwidth on their interfaces.

Without IPPCP software, compression is not supported with the VPN encryption hardware AIM and NM; that is, a user has to remove the VPN module from the router and run the software encryption with software compression. IPPCP enables all VPN modules to support LZS compression in the software when the VPN module is in the router, thereby allowing users to configure data compression and increase their bandwidth, which is useful for a low data link.

Without IPPCP, compression occurs at Layer 2, and encryption occurs at Layer 3. After a data stream is encrypted, it is passed on for compression services. When the compression engine receives the encrypted data streams, the data expands and does not compress. This feature enables both compression and encryption of the data to occur at Layer 3 by selecting LZS with the IPsec transform set; that is, LZS compression occurs before encryption, and with better compression ratio.

Table 1 AIM/VPN Encryption Module Support by Cisco IOS Release

Platform	Encryption Module Support by Cisco IOS Release—12.2(13)T	Encryption Module Support by Cisco IOS Release—12.3(4)T	Encryption Module Support by Cisco IOS Release—12.3(5)	Encryption Module Support by Cisco IOS Release—12.3(6)	Encryption Module Support by Cisco IOS Release—12.3(7)T
Cisco 831	Software-based AES	Software-based AES	Software-based AES	Software-based AES	Software-based AES

Platform	Encryption Module Support by Cisco IOS Release—12.2(13)T	Encryption Module Support by Cisco IOS Release—12.3(4)T	Encryption Module Support by Cisco IOS Release—12.3(5)	Encryption Module Support by Cisco IOS Release—12.3(6)	Encryption Module Support by Cisco IOS Release—12.3(7)T
Cisco 1710 Cisco 1711 Cisco 1721 Cisco 1751 Cisco 1760	Software-based AES	Software-based AES	Software-based AES	Software-based AES	Software-based AES
Cisco 2600 XM	—	—	—	AIM-VPN/ BPII-Plus Hardware Encryption Module	AIM-VPN/ BPII-Plus Hardware Encryption Module
Cisco 2611 XM Cisco 2621 XM Cisco 2651 XM	—	AIM-VPN/BPII Hardware Encryption Module	AIM-VPN/BPII Hardware Encryption Module	AIM-VPN/BPII Hardware Encryption Module	AIM-VPN/ BPII-Plus Hardware Encryption Module
Cisco 2691 XM	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/ EPII-Plus Hardware Encryption Module
Cisco 3660 Cisco 3745	AIM-VPN/HPII Hardware Encryption Module	AIM-VPN/HPII Hardware Encryption Module	AIM-VPN/ HPII-Plus Hardware Encryption Module	AIM-VPN/ HPII-Plus Hardware Encryption Module	AIM-VPN/ HPII-Plus Hardware Encryption Module
Cisco 3735	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/EPII Hardware Encryption Module	AIM-VPN/ EPII-Plus Hardware Encryption Module	AIM-VPN/ EPII-Plus Hardware Encryption Module	AIM-VPN/ EPII-Plus Hardware Encryption Module

For more information on AIMs and NM, see the *Installing Advanced Integration Modules in Cisco 2600 Series, Cisco 3600 Series, and Cisco 3700 Series Routers* document.

Supported Switching Paths

The table below lists the supported switching paths that work with IPsec.

Table 2 Supported Switching Paths for IPsec

Switching Paths	Examples

Cisco Express Forwarding	`ip cef`
	`interface ethernet0/0`
	`ip route-cache`
	`! Ensure that you will not hit flow switching.`
	`no ip route-cache flow`

Cisco Express Forwarding-flow switching	`! Enable global CEF.`
	`ip cef`
	`interface ethernet0/0`
	`ip route-cache`
	`ip route-cache flow`
	`! Enable CEF for the interface`
	`ip route-cache cef`

Fast switching	`interface ethernet0/0`
	`ip route-cache`
	`! Ensure that you will not hit flow switching.`
	`no ip route-cache flow`
	`! Disable CEF for the interface, which supersedes global CEF.`
	`no ip route-cache cef`

Fast-flow switching	`interface ethernet0/0`
	`ip route-cache`
	`! Enable flow switching`
	`p route-cache flow`
	`! Disable CEF for the interface.`
	`no ip route-cache cef`

Process switching	`interface ethernet0/0` `no ip route-cache`

Supported Encapsulation

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

IPsec also works with Generic Routing Encapsulation (GRE) and IPinIP Layer 3, Layer 2 Forwarding (L2F), Layer 2 Tunneling Protocol (L2TP), Data Link Switching+ (DLSw+), and Source Route Bridging (SRB) tunneling protocols; however, multipoint tunnels are not supported. 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, the 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 sent IPsec packets. 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 the characteristics of these tunnels. When the IPsec peer recognizes 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 the protocols and algorithms to 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 can define the traffic that needs to be protected between two IPsec peers by configuring access lists and applying these access lists to interfaces using crypto map sets. Therefore, traffic may be selected on the basis of the source and destination address, and optionally the Layer 4 protocol and port. (The access lists used for IPsec are only used to determine the traffic that needs to be protected by IPsec, not the traffic that 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 a sequence—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 there is no SA that the 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 33 section later in this module.)

If the crypto map entry is tagged as **ipsec-manual**, IPsec is triggered. If there is no SA 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 SAs do not exist, IPsec does not have all 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 only need to be authenticated, while other data streams must both be encrypted and authenticated.

Access lists associated with IPsec crypto map entries also represent the traffic that the router needs protected by IPsec. Inbound traffic is processed against 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 that can be applied to IPsec-protected traffic. During the IPsec SA negotiation, the peers agree to use a particular transform set when protecting a particular data flow.

- IKEv1 Transform Sets, page 9
- IKEv2 Transform Sets, page 9

IKEv1 Transform Sets

An Internet Key Exchange version 1 (IKEv1) 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, peers search for a transform set that is the same at both peers. When such a transform set is found, it is selected and 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 applied only 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.

IKEv2 Transform Sets

An Internet Key Exchange version 2 (IKEv2) proposal is a set of transforms used in the negotiation of IKEv2 SA as part of the IKE_SA_INIT exchange. An IKEv2 proposal is regarded as complete only when it has at least an encryption algorithm, an integrity algorithm, and a Diffie-Hellman (DH) group configured. If no proposal is configured and attached to an IKEv2 policy, then the default proposal is used in the negotiation. The default proposal is a collection of commonly used algorithms which are as follows:

```
encryption aes-cbc-128 3des
integrity sha md5
group 5 2
```

The transforms shown above translate to the following combinations in the following order of priority:

```
aes-cbc-128, sha, 5
aes-cbc-128, sha, 2
aes-cbc-128, md5, 5
aes-cbc-128, md5, 2
3des, sha1, 5
3des, sha1, 2
3des, md5, 5
3des, md5, 2
```

Although the **crypto ikev2 proposal** command is similar to the **crypto isakmp policy priority** command, the IKEv2 proposal differs as follows:

- An IKEv2 proposal allows configuration of one or more transforms for each transform type.
- An IKEv2 proposal does not have any associated priority.

Note To use IKEv2 proposals in negotiation, they must be attached to IKEv2 policies. If a proposal is not configured, then the default IKEv2 proposal is used with the default IKEv2 policy.

When multiple transforms are configured for a transform type, the order of priority is from left to right.

A proposal with multiple transforms for each transform type translates to all possible combinations of transforms. If only a subset of these combinations is required, then they must be configured as individual proposals.

```
Device(config)# crypto ikev2 proposal proposal-1
Device(config-ikev2-proposal)# encryption aes-cbc-128, aes-cbc-192
Device(config-ikev2-proposal)# integrity sha1, sha256
Device(config-ikev2-proposal)# group 14
```

For example, the commands shown above translate to the following transform combinations:

```
aes-cbc-128, sha1, 14
aes-cbc-192, sha1, 14
aes-cbc-128, sha256, 14
aes-cbc-192, sha256, 14
```

To configure the first and last transform combinations, use the following commands:

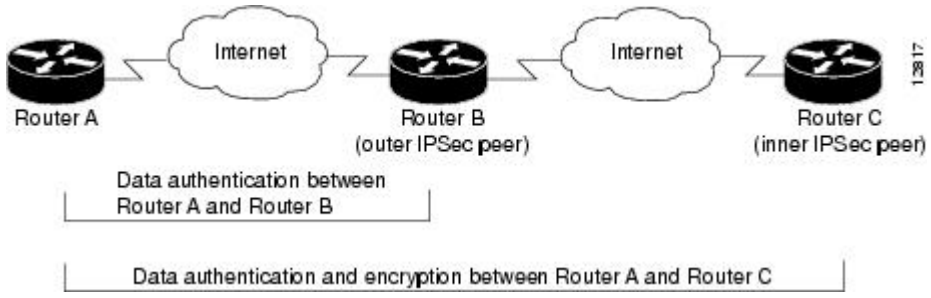
```
Device(config)# crypto ikev2 proposal proposal-1
Device(config-ikev2-proposal)# encryption aes-cbc-128
Device(config-ikev2-proposal)# integrity sha1
Device(config-ikev2-proposal)# group 14
Device(config-ikev2-proposal)# exit
Device(config)# crypto ikev2 proposal proposal-2
Device(config-ikev2-proposal)# encryption aes-cbc-192
Device(config-ikev2-proposal)# integrity sha256
Device(config-ikev2-proposal)# group 14
```

IPsec Traffic Nested to Multiple Peers

You can nest IPsec traffic to a series of IPsec peers. For example, 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 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 a different protection (such as both data authentication and encryption).

Crypto Access Lists

- Crypto Access List Overview, page 11

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 four primary functions:

- Select outbound traffic to be protected by IPsec (permit = protect).
- Indicate 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 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.
- Perform negotiation only for **ipsec-isakmp** crypto map entries.

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 the 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 criteria is applied in the forward direction to traffic exiting your router and in the reverse direction to traffic entering your router.

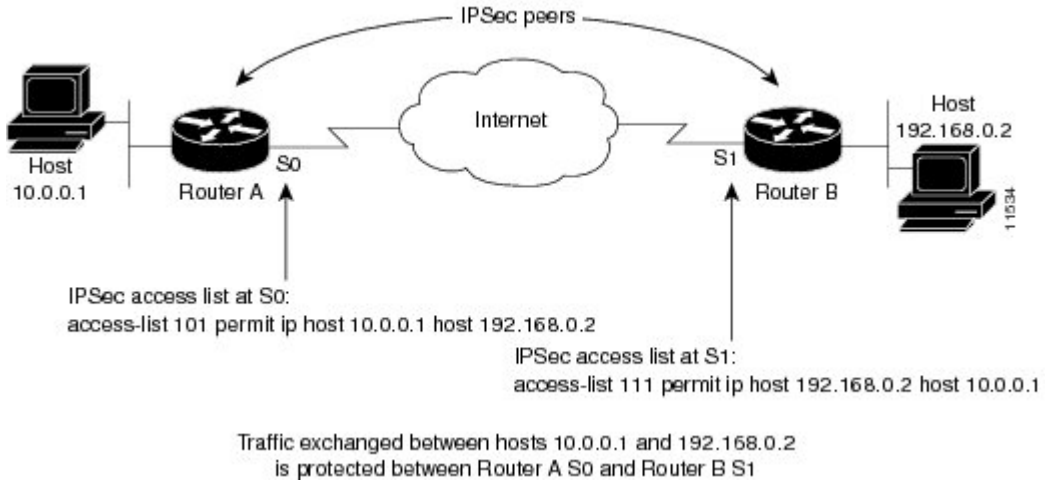
In the figure below, IPsec protection is applied to traffic between Host 10.0.0.1 and Host 192.168.0.2 as the data exits Router A's S0 interface en route to Host 192.168.0.2. For traffic from Host 10.0.0.1 to Host 192.168.0.2, the access list entry on Router A is evaluated as follows:

```
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, the 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



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 a **deny** keyword in one crypto map sequence number and having a **permit** keyword for the same subnet and IP range in another crypto map sequence number are not supported.

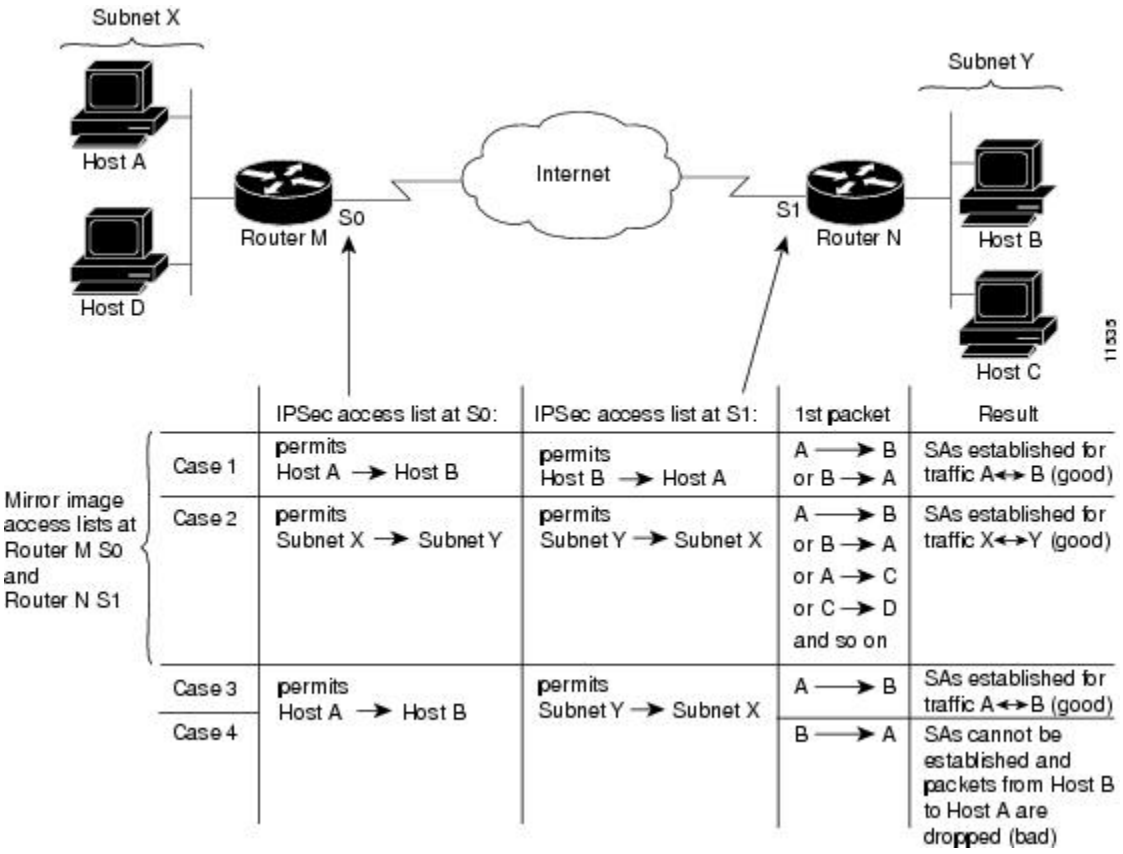
```
crypto map mymap 10 ipsec-isakmp
 set peer 192.168.1.1
 set transform-set test
 match address 101
crypto map mymap 20 ipsec-isakmp
 set peer 192.168.1.2
 set transform-set test
 match address 102
access-list 101 deny   ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 101 permit ip 10.0.0.0 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 of mirror image and nonmirror image access lists.

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



As the above figure 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 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 in the above figure. IPsec SA establishment is critical to IPsec—without SAs, IPsec does not work, thus causing 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, so 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 the source or destination addresses. By default, VTI solutions use the **any** keyword as a proxy identity. Use of VTI is encouraged when proxy identities of **any** are required.

The **any** keyword in a **permit** statement is not recommended when you have multicast traffic flowing through the IPsec interface; the **any** keyword can cause multicast traffic to fail.

Note In Cisco IOS Release 12.4(9)T and later releases, multicast traffic from the router will be encapsulated into IPsec if proxy identities allow encapsulation.

The **permit any any** statement is strongly discouraged because this causes all outbound traffic to be protected (and all protected traffic is 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, the Network Time Protocol (NTP), echo, echo response, and so on.

You need to be sure that 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.

The use of the **any** keyword in access control lists (ACLs) with reverse route injection (RRI) is not supported. (For more information on RRI, see the section "Creating Crypto Map Sets, page 30.")

Transform Sets: A Combination of Security Protocols and Algorithms

About Transform Sets

Note Cisco no longer recommends using ah-md5-hmac, esp-md5-hmac, esp-des or esp-3des. Instead, you should use ah-sha-hmac, esp-sha-hmac or esp-aes. For more information about the latest Cisco cryptographic recommendations, see the *Next Generation Encryption* (NGE) white paper.

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, peers search for an identical transform set for both peers. When such a transform set is found, it is selected and 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 3 Allowed Transform Combinations

Transform Type	Transform	Description
AH Transform (*Pick only one.*)	**ah-md5-hmac**	AH with the MD5 (Message Digest 5) (an HMAC variant) authentication algorithm. (No longer recommened).
	ah-sha-hmac	AH with the SHA (Secure Hash Algorithm) (an HMAC variant) authentication algorithm.
ESP Encryption Transform (*Pick only one.*)	**esp-aes**	ESP with the 128-bit Advanced Encryption Standard (AES) encryption algorithm.
	esp-gcm **esp-gmac**	The **esp-gcm** and **esp-gmac** transforms are ESPs with either a 128-bit or a 256-bit encryption algorithm. The default for either of these transforms is 128 bits. Both **esp-gcm** and **esp-gmac** transforms cannot be configured together with any other ESP transform within the same crypto IPsec transform set using the **crypto ipsec transform-set** command.

Transform Type	Transform	Description
	esp-aes 192	ESP with the 192-bit AES encryption algorithm.
	esp-aes 256	ESP with the 256-bit AES encryption algorithm.
	esp-des	ESP with the 56-bit Data Encryption Standard (DES) encryption algorithm. (No longer recommended).
	esp-3des	ESP with the 168-bit DES encryption algorithm (3DES or Triple DES). (No longer recommended).
	esp-null	Null encryption algorithm.
	esp-seal	ESP with the 160-bit SEAL encryption algorithm.
ESP Authentication Transform (*Pick only one.*)	**esp-md5-hmac**	ESP with the MD5 (HMAC variant) authentication algorithm. (No longer recommended).
	esp-sha-hmac	ESP with the SHA (HMAC variant) authentication algorithm.
IP Compression Transform	**comp-lzs**	IP compression with the Lempel-Ziv-Stac (LZS) algorithm

Cisco IOS Suite-B Support for IKE and IPsec Cryptographic Algorithms

Suite-B adds support for four user interface suites of cryptographic algorithms for use with IKE and IPSec that are described in RFC 4869. Each suite consists of an encryption algorithm, a digital signature algorithm, a key agreement algorithm, and a hash or message digest algorithm.

Suite-B has the following cryptographic algorithms:

- Suite-B-GCM-128-Provides ESP integrity protection, confidentiality, and IPsec encryption algorithms that use the 128-bit AES using Galois and Counter Mode (AES-GCM) described in RFC 4106. This suite should be used when ESP integrity protection and encryption are both needed.
- Suite-B-GCM-256-Provides ESP integrity protection and confidentiality using 256-bit AES-GCM described in RFC 4106. This suite should be used when ESP integrity protection and encryption are both needed.
- Suite-B-GMAC-128-Provides ESP integrity protection using 128-bit AES- Galois Message Authentication Code (GMAC) described in RFC 4543, but does not provide confidentiality. This suite should be used only when there is no need for ESP encryption.
- Suite-B-GMAC-256-Provides ESP integrity protection using 256-bit AES-GMAC described in RFC 4543, but does not provide confidentiality. This suite should be used only when there is no need for ESP encryption.

IPSec encryption algorithms use AES-GCM when encryption is required and AES-GMAC for message integrity without encryption.

IKE negotiation uses AES Cipher Block Chaining (CBC) mode to provide encryption and Secure Hash Algorithm (SHA)-2 family containing the SHA-256 and SHA-384 hash algorithms, as defined in RFC 4634, to provide the hash functionality. Diffie-Hellman using Elliptic Curves (ECP), as defined in RFC 4753, is used for key exchange and the Elliptic Curve Digital Signature Algorithm (ECDSA), as defined in RFC 4754, to provide authentication.

Suite-B Requirements

Suite-B imposes the following software crypto engine requirements for IKE and IPsec:

- HMAC-SHA256 and HMAC-SHA384 are used as pseudorandom functions; the integrity check within the IKE protocol is used. Optionally, HMAC-SHA512 can be used.
- Elliptic curve groups 19 (256-bit ECP curve) and 20 (384-bit ECP curve) are used as the Diffie-Hellman group in IKE. Optionally, group 21 (521-bit ECP curve) can be used.
- The Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm (256-bit and 384-bit curves) is used for the signature operation within X.509 certificates.
- GCM (16 byte ICV) and GMAC is used for ESP (128-bit and 256-bit keys). Optionally, 192-bit keys can be used.
- Public Key Infrastructure (PKI) support for validation of X.509 certificates using ECDSA signatures must be used.
- PKI support for generating certificate requests using ECDSA signatures and for importing the issued certificates into IOS must be used.
- IKEV2 support for allowing the ECDSA signature (ECDSA-sig) as authentication method must be used.

Where to Find Suite-B Configuration Information

Suite-B configuration support is described in the following documents:

- For more information on the **esp-gcm** and **esp-gmac** transforms, see the "Configuring Transform Sets for IKEv1, page 26" section.
- For more information on SHA-2 family (HMAC variant) and Elliptic Curve (EC) key pair configuration, see the *Configuring Internet Key Exchange for IPsec VPNs* feature module.
- For more information on configuring a transform for an integrity algorithm type, see the "Configuring the IKEv2 Proposal" section in the *Configuring Internet Key Exchange Version 2 (IKEv2) and FlexVPN Site-to-Site* feature module.
- For more information on configuring the ECDSA-sig to be the authentication method for IKEv2, see the "Configuring IKEv2 Profile (Basic)" section in the *Configuring Internet Key Exchange Version 2 (IKEv2) and FlexVPN Site-to-Site* feature module.
- For more information on configuring elliptic curve Diffie-Hellman (ECDH) support for IPsec SA negotiation, see the *Configuring Internet Key Exchange for IPsec VPNs* and *Configuring Internet Key Exchange Version 2 and FlexVPN* feature modules.

For more information on the Suite-B support for certificate enrollment for a PKI, see the *Configuring Certificate Enrollment for a PKI* feature module.

Crypto Map Sets

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

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 "Applying Crypto Map Sets to Interfaces, page 40" section 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; 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, as well as 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 meet the following criteria:

- The crypto map entries must contain compatible crypto access lists (for example, mirror image access lists). In cases, 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 load sharing. Load sharing is useful because if one peer fails, there is still a protected path. The peer to which packets are actually sent is determined by the last peer that the router heard from (that is, received either traffic or a negotiation request) 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 33”. 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.

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 first evaluated against the higher priority map entries.

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 (for 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 such cases, the different types of traffic should be 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 you 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.

Static Crypto Maps

When IKE is used to establish SAs, 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. To create IPv6 crypto map entries, you must use the **ipv6** keyword with the **crypto map** command. For IPv4 crypto maps, use the **crypto map** command without the **ipv6** keyword.

Dynamic Crypto Maps

Dynamic crypto maps can simplify IPsec configuration and are recommended for use with networks where the peers are not always predetermined. To create dynamic crypto maps, you should understand the following concepts:

Dynamic Crypto Maps Overview

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, it installs the new IPsec security associations, it also installs a temporary crypto map entry. This entry contains 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 corresponding security associations expire), the temporary crypto map entry is 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," 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 exists, the traffic would simply be dropped (because dynamic crypto maps are not used for initiating new SAs).

Note Use care when using the **any** keyword in **permit** entries in dynamic crypto maps. If the traffic covered by such a **permit** entry can 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.

Tunnel Endpoint Discovery

Defining a dynamic crypto map allows only the receiving router to dynamically determine an IPsec peer. Tunnel Endpoint Discovery (TED) allows the initiating router to dynamically determine an IPsec peer for secure IPsec communications.

Dynamic TED helps to simplify the IPsec configuration on individual routers within a large network. Each node has a simple configuration that defines the local network that the router is protecting and the required IPsec transforms.

To have a large, fully-meshed network without TED, each peer needs to have static crypto maps to every other peer in the network. For example, if there are 100 peers in a large, fully-meshed network, each router needs 99 static crypto maps for each of its peers. With TED, only a single dynamic TED-enabled crypto map is needed because the peer is discovered dynamically. Thus, static crypto maps do not need to be configured for each peer.

Note TED only helps in discovering peers and does not function any differently than normal IPsec. TED does not improve the scalability of IPsec (in terms of performance or the number of peers or tunnels).

The figure below and the corresponding steps explain a sample TED network topology.

Figure 4 Tunnel Endpoint Discovery Sample Network Topology

SUMMARY STEPS

1. Host A sends a packet that is destined for Host B.
2. Router 1 intercepts and reads the packet. According to the IKE policy, Router 1 contains the following information: the packet must be encrypted, there are no SAs for the packet, and TED is enabled. Thus, Router 1 drops the packet and sends a TED probe into the network. (The TED probe contains the IP address of Host A (as the source IP address) and the IP address of Host B (as the destination IP address) embedded in the payload.
3. Router 2 intercepts the TED probe and checks the probe against the ACLs that it protects; after the probe matches an ACL, it is recognized as a TED probe for proxies that the router protects. The probe then sends a TED reply with the IP address of Host B (as the source IP address) and the IP address of Host A (as the destination IP address) embedded in the payload.
4. Router 1 intercepts the TED reply and checks the payloads for the IP address and half proxy of Router 2. It then combines the source side of its proxy with the proxy found in the second payload and initiates an IKE session with Router 2; thereafter, Router 1 initiates an IPsec session with Router 2.

DETAILED STEPS

Step 1 Host A sends a packet that is destined for Host B.

Step 2 Router 1 intercepts and reads the packet. According to the IKE policy, Router 1 contains the following information: the packet must be encrypted, there are no SAs for the packet, and TED is enabled. Thus, Router 1 drops the packet

and sends a TED probe into the network. (The TED probe contains the IP address of Host A (as the source IP address) and the IP address of Host B (as the destination IP address) embedded in the payload.

Step 3 Router 2 intercepts the TED probe and checks the probe against the ACLs that it protects; after the probe matches an ACL, it is recognized as a TED probe for proxies that the router protects. The probe then sends a TED reply with the IP address of Host B (as the source IP address) and the IP address of Host A (as the destination IP address) embedded in the payload.

Step 4 Router 1 intercepts the TED reply and checks the payloads for the IP address and half proxy of Router 2. It then combines the source side of its proxy with the proxy found in the second payload and initiates an IKE session with Router 2; thereafter, Router 1 initiates an IPsec session with Router 2.

Note IKE cannot occur until the peer is identified.

TED Versions

The following table lists the available TED versions:

Version	First Available Release	Description
TEDv1	12.0(5)T	Performs basic TED functionality on nonredundant networks.
TEDv2	12.1M	Enhanced to work with redundant networks with paths through multiple security gateways between the source and the destination.
TEDv3	12.2M	Enhanced to allow non-IP-related entries to be used in the access list.

TED Restrictions

TED has the following restrictions:

- It is Cisco proprietary.
- It is available only on dynamic crypto maps. (The dynamic crypto map template is based on the dynamic crypto map performing peer discovery. Although there are no access-list restrictions on the dynamic crypto map template, the dynamic crypto map template should cover data sourced from the protected traffic and the receiving router using the **any** keyword. When using the **any** keyword, include explicit **deny** statements to exempt routing protocol traffic prior to entering the **permit any** command.
- TED works only in tunnel mode; that is, it does not work in transport mode.
- It is limited by the performance and scalability of the limitation of IPsec on each individual platform.

Note Enabling TED slightly decreases the general scalability of IPsec because of the set-up overhead of peer discovery, which involves an additional “round-trip” of IKE messages (TED probe and reply). Although minimal, the additional memory used to store data structures during the peer discovery stage adversely affects the general scalability of IPsec.

- IP addresses must be routed within the network.
- The access list used in the crypto map for TED can only contain IP-related entries—TCP, UDP, or other protocols cannot be used in the access list.

Note This restriction is no longer applicable in TEDv3.

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 must 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.

Establish 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 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 support multiple manually established SAs for different kinds of traffic, define multiple crypto access lists, and 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 you want to create.

DETAILED STEPS

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

	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:
	`Device(config)# access-list 100 permit ip 10.0.68.0 0.0.0.255 10.1.1.0 0.0.0.255`
	Example:
	`Device(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 you want to create.	—

• What to Do Next, page 25

What to Do Next

After at least one crypto access list is created, a transform set needs to be defined as described in the "Configuring Transform Sets for IKEv1 and IKEv2 Proposals, page 25" section.

Next the crypto access lists need to be associated to particular interfaces when you configure and apply crypto map sets to the interfaces. (Follow the instructions in the "Creating Crypto Map Sets, page 30" and "Applying Crypto Map Sets to Interfaces, page 40" sections).

Configuring Transform Sets for IKEv1 and IKEv2 Proposals

Perform this task to define a transform set that is to be used by the IPsec peers during IPsec security association negotiations with IKEv1 and IKEv2 proposals.

Restrictions

If you are specifying SEAL encryption, note the following restrictions:

- Your router and the other peer must not have a hardware IPsec encryption.
- Your router and the other peer must support IPsec.
- Your router and the other peer must support the k9 subsystem.
- SEAL encryption is available only on Cisco equipment. Therefore, interoperability is not possible.
- Unlike IKEv1, the authentication method and SA lifetime are not negotiable in IKEv2, and because of this, these parameters cannot be configured under the IKEv2 proposal.

Configuring Transform Sets for IKEv1

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec transform-set** *transform-set-name transform1* [*transform2* [*transform3*]]
4. **mode** [**tunnel** | **transport**]
5. **end**
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:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec transform-set** *transform-set-name transform1* [*transform2* [*transform3*]] **Example:** `Device(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 the entries that you can use for transform arguments. These rules are explained in the command description for the **crypto ipsec transform-set** command, and the table in "About Transform Sets, page 14" section provides a list of allowed transform combinations.
Step 4	**mode** [**tunnel**	**transport**] **Example:** `Device(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	**end** **Example:** `Device(cfg-crypto-tran)# end`	Exits crypto transform configuration mode and enters privileged EXEC mode.

	Command or Action	Purpose
Step 6	**clear crypto sa** [**peer** {*ip-address* | *peer-name*} | **sa map** *map-name* | **sa entry** *destination-address protocol spi*] **Example:** `Device# 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 clears 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:** `Device# show crypto ipsec transform-set`	(Optional) Displays the configured transform sets.

• What to Do Next, page 27

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, page 30" section.

Configuring Transform Sets for IKEv2

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ikev2 proposal** *proposal-name*
4. **encryption** *transform1* [*transform2*] ...
5. **integrity** *transform1* [*transform2*] ...
6. **group** *transform1* [*transform2*] ...
7. **end**
8. **show crypto ikev2 proposal**

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ikev2 proposal** *proposal-name* **Example:** `Device(config)# crypto ikev2 proposal proposal-1`	Specifies the name of the proposal and enters crypto IKEv2 proposal configuration mode. • The proposals are referred in IKEv2 policies through the proposal name.
Step 4	**encryption** *transform1* [*transform2*] ... **Example:** `Device(config-ikev2-proposal)# encryption aes-cbc-128`	(Optional) Specifies one or more transforms of the following encryption type: • AES-CBC 128—128-bit AES-CBC • AES-CBC 192—192-bit AES-CBC • AES-CBC 256—256-bit AES-CBC • 3DES—168-bit DES (No longer recommended. AES is the recommended encryption algorithm).
Step 5	**integrity** *transform1* [*transform2*] ... **Example:** `Device(config-ikev2-proposal)# integrity sha1`	(Optional) Specifies one or more transforms of the following integrity type: • The **sha256** keyword specifies SHA-2 family 256-bit (HMAC variant) as the hash algorithm. • The **sha384** keyword specifies SHA-2 family 384-bit (HMAC variant) as the hash algorithm. • The **sha512** keyword specifies SHA-2 family 512-bit (HMAC variant) as the hash algorithm • the **sha1** keyword specifies the SHA-1 (HMAC variant) as the hash algorithm. • The **md5** keyword specifies MD5 (HMAC variant) as the hash algorithm. (No longer recommended. SHA-1 is the recommended replacement.)
Step 6	**group** *transform1* [*transform2*] ... **Example:** `Device(config-ikev2-proposal)# group 14`	(Optional) Specifies one or more transforms of the possible DH group type: • **1**—768-bit DH (No longer recommended.) • **2**—1024-bit DH (No longer recommended) • **5**—1536-bit DH (No longer recommended) • **14**—Specifies the 2048-bit DH group. • **15**—Specifies the 3072-bit DH group. • **16**—Specifies the 4096-bit DH group. • **19**—Specifies the 256-bit elliptic curve DH (ECDH) group. • **20**—Specifies the 384-bit ECDH group. • **24**—Specifies the 2048-bit DH/DSA group.

	Command or Action	Purpose
Step 7	**end** **Example:** `Device(config-ikev2-proposal)# end`	Exits crypto IKEv2 proposal configuration mode and returns to privileged EXEC mode.
Step 8	**show crypto ikev2 proposal** **Example:** `Device# show crypto ikev2 proposal`	(Optional) Displays the parameters for each IKEv2 proposal.

- Transform Sets for IKEv2 Examples, page 29
- What to Do Next, page 30

Transform Sets for IKEv2 Examples

The following examples show how to configure a proposal:

IKEv2 Proposal with One Transform for Each Transform Type

```
Device(config)# crypto ikev2 proposal proposal-1
Device(config-ikev2-proposal)# encryption aes-cbc-128
Device(config-ikev2-proposal)# integrity sha1
Device(config-ikev2-proposal)# group 14
```

IKEv2 Proposal with Multiple Transforms for Each Transform Type

```
Device(config)# crypto ikev2 proposal proposal-2
Device(config-ikev2-proposal)# encryption aes-cbc-128 aes-cbc-192
Device(config-ikev2-proposal)# integrity sha2 sha256
Device(config-ikev2-proposal)# group 14 15
```

The IKEv2 proposal **proposal-2** translates to the following prioritized list of transform combinations:

- aes-cbc-128, sha1, 14
- aes-cbc-128, sha1, 15
- aes-cbc-128, sha256, 14
- aes-cbc-128, sha256, 15
- aes-cbc-192, sha1, 14
- aes-cbc-192, sha1, 15
- aes-cbc-192, sha256, 14
- aes-cbc-192, sha256, 15

IKEv2 Proposals on the Initiator and Responder

The proposal of the initiator is as follows:

```
Device(config)# crypto ikev2 proposal proposal-1
Device(config-ikev2-proposal)# encryption aes-cbc-128 aes-cbc-196
```

```
Device(config-ikev2-proposal)# integrity sha1 sha256
Device(config-ikev2-proposal)# group 14 16
```

The proposal of the responder is as follows:

```
Device(config)# crypto ikev2 proposal proposal-2
Device(config-ikev2-proposal)# encryption aes-cbc-196 aes-cbc-128
Device(config-ikev2-proposal)# integrity sha256 sha1
Device(config-ikev2-proposal)# group 16 14
```

In the scenario, the initiator's choice of algorithms is preferred and the selected algorithms are as follows:

```
encryption aes-cbc-128
integrity sha1
group 14
```

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, page 30" section.

Creating Crypto Map Sets

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 SAs. To create IPv6 crypto map entries, you must use the **ipv6** keyword with the **crypto map** command. For IPv4 crypto maps, use the **crypto map** command without the **ipv6** keyword.

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

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto map** [**ipv6**] *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* | **kilobytes disable**}
8. **set security-association level per-host**
9. **set pfs** [**group1** | **group14** | **group15** | **group16** | **group19** | **group2** | **group20** | **group24 | group5**]
10. **end**
11. **show crypto map** [**interface** *interface* | **tag** *map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto map** [**ipv6**] *map-name seq-num* [**ipsec-isakmp**] **Example:** `Device(config)# crypto map static-map 1 ipsec-isakmp`	Creates or modifies a crypto map entry, and enters crypto map configuration mode. • For IPv4 crypto maps, use the command without the **ipv6** keyword.
Step 4	**match address** *access-list-id* **Example:** `Device(config-crypto-m)# match address vpn-tunnel`	Names an extended access list. • This access list determines the traffic that should be protected by IPsec and the traffic that should not be protected by IPsec security in the context of this crypto map entry.
Step 5	**set peer** {*hostname*	*ip-address*} **Example:** `Device(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.

	Command or Action	Purpose
Step 6	**set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*] **Example:** `Device(config-crypto-m)# set transform-set aesset`	Specifies the transform sets that are allowed for this crypto map entry. • List multiple transform sets in the order of priority (highest priority first).
Step 7	**set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes* | **kilobytes disable**} **Example:** `Device (config-crypto-m)# set security-association lifetime seconds 2700`	(Optional) Specifies a SA lifetime for the crypto map entry. • By default, the SAs of the crypto map are negotiated according to the global lifetimes, which can be disabled.
Step 8	**set security-association level per-host** **Example:** `Device(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.
Step 9	**set pfs** [**group1** | **group14** | **group15** | **group16** | **group19** | **group2** | **group20** | **group24** | **group5**] **Example:** `Device(config-crypto-m)# set pfs group14`	(Optional) Specifies that IPsec either should ask for password forward secrecy (PFS) when requesting new SAs for this crypto map entry or should demand PFS in requests received from the IPsec peer. • Group 1 specifies the 768-bit Diffie-Hellman (DH) identifier (default). (No longer recommended). • Group 2 specifies the 1024-bit DH identifier. (No longer recommended). • Group 5 specifies the 1536-bit DH identifier. (No longer recommended) • Group 14 specifies the 2048-bit DH identifier. • Group 15 specifies the 3072-bit DH identifier. • Group 16 specifies the 4096-bit DH identifier. • Group 19 specifies the 256-bit elliptic curve DH (ECDH) identifier. • Group 20 specifies the 384-bit ECDH identifier. • Group 24 specifies the 2048-bit DH/DSA identifier • By default, PFS is not requested. If no group is specified with this command, group 1 is used as the default.

	Command or Action	Purpose
Step 10	**end** **Example:** `Device(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:** `Device# show crypto map`	Displays your crypto map configuration.

- Troubleshooting Tips, page 33
- What to Do Next, page 33

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, 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 "Applying Crypto Map Sets to Interfaces, page 40" section.

Creating Dynamic Crypto Maps

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 the acceptable transform sets.

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

Note IPv6 addresses are not supported on dynamic crypto maps.

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

SUMMARY STEPS

1. **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* | **kilobytes disable**}
8. **set pfs** [**group1** | **group14** | **group15** | **group16** | **group19** | **group2** | **group20** | **group24** | **group5**]
9. **exit**
10. **exit**
11. **show crypto dynamic-map** [**tag** *map-name*]
12. **configure terminal**
13. **crypto map** *map-name seq-num* **ipsec-isakmp dynamic** *dynamic-map-name* [**discover**]
14. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto dynamic-map** *dynamic-map-name dynamic-seq-num* **Example:** `Device(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:** `Device(config-crypto-m)# set transform-set aesset`	Specifies the transform sets allowed for the crypto map entry. • List multiple transform sets in the 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:** `Device(config-crypto-m)# match address 101`	(Optional) Specifies the 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 an access list is configured, the data flow identity proposed by the IPsec peer must fall within a **permit** statement for this crypto access list. • If an access list is not configured, the device accepts any data flow identity proposed by the IPsec peer. However, if an access list is configured but the specified access list does not exist or is empty, the device drops all packets. This is similar to static crypto maps, which require access lists to 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, and you cannot enable TED because packets are sent in the clear (unencrypted.)
Step 6	**set peer** {*hostname* | *ip-address*} **Example:** `Device(config-crypto-m)# set peer 192.168.101.1`	(Optional) Specifies a remote IPsec peer. Repeat this step 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* | **kilobytes disable**} **Example:** `Device(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 IP Security SAs. **Note** To minimize the possibility of packet loss when rekeying in high bandwidth environments, you can disable the rekey request triggered by a volume lifetime expiry.

	Command or Action	Purpose
Step 8	**set pfs** [**group1** | **group14** | **group15** | **group16** | **group19** | **group2** | **group20** | **group24** | **group5**] **Example:** `Device(config-crypto-m)# set pfs group14`	(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. • Group 1 specifies the 768-bit Diffie-Hellman (DH) identifier (default). (No longer recommended). • Group 2 specifies the 1024-bit DH identifier. (No longer recommended). • Group 5 specifies the 1536-bit DH identifier. (No longer recommended) • Group 14 specifies the 2048-bit DH identifier. • Group 15 specifies the 3072-bit DH identifier. • Group 16 specifies the 4096-bit DH identifier. • Group 19 specifies the 256-bit elliptic curve DH (ECDH) identifier. • Group 20 specifies the 384-bit ECDH identifier. • Group 24 specifies the 2048-bit DH/DSA identifier • By default, PFS is not requested. If no group is specified with this command, **group1** is used as the default.
Step 9	**exit** **Example:** `Device(config-crypto-m)# exit`	Exits crypto map configuration mode and returns to global configuration mode.
Step 10	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 11	**show crypto dynamic-map** [**tag** *map-name*] **Example:** `Device# show crypto dynamic-map`	(Optional) Displays information about dynamic crypto maps.
Step 12	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 13	**crypto map** *map-name seq-num* **ipsec-isakmp dynamic** *dynamic-map-name* [**discover**] **Example:** `Device(config)# crypto map static-map 1 ipsec-isakmp dynamic test-map discover`	(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. **Note** You must enter the **discover** keyword to enable TED.
Step 14	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.

- Troubleshooting Tips, page 37
- What to Do Next, page 37

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, 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 entire SA database must be reserved for large-scale changes, or when the router is processing minimal 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 "Applying Crypto Map Sets to Interfaces, page 40" section.

Creating Crypto Map Entries to Establish Manual SAs

Perform this task to create crypto map entries to establish manual SAs (that is, when IKE is not used to establish the SAs). To create IPv6 crypto maps entries, you must use the **ipv6** keyword with the **crypto map** command. For IPv4 crypto maps, use the **crypto map** command without the **ipv6** keyword.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto map** [**ipv6**] *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. Do one of the following:
   - **set session-key inbound ah** *spi hex-key-string*
   - **set session-key outbound ah** *spi hex-key-string*
8. Do one of the following:
   - **set session-key inbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
   - **set session-key outbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
9. **exit**
10. **exit**
11. **show crypto map** [**interface** *interface* | **tag** *map-name*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto map** [**ipv6**] *map-name seq-num* [**ipsec-manual**] **Example:** `Device(config)# crypto map mymap 10 ipsec-manual`	Specifies the crypto map entry to be created or modified and enters crypto map configuration mode. • For IPv4 crypto maps, use the **crypto map** command without the **ipv6** keyword.
Step 4	**match address** *access-list-id* **Example:** `Device(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:** `Device(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:** `Device(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	Do one of the following:
	• **set session-key inbound ah** *spi hex-key-string*
	• **set session-key outbound ah** *spi hex-key-string*
	
	Example:
	`Device(config-crypto-m)# set session-key inbound ah 256 9876543210987654987654321098765 4`
	
	Example:
	`Device(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	Do one of the following:
	• **set session-key inbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
	• **set session-key outbound esp** *spi* **cipher** *hex-key-string* [**authenticator** *hex-key-string*]
	
	Example:
	`Device(config-crypto-m)# set session-key inbound esp 256 cipher 0123456789012345`
	
	Example:
	`Device(config-crypto-m)# set session-key outbound esp 256 cipher abcdefabcdefabcd`
		Sets the Encapsulating Security Payload (ESP) Security Parameter Indexes (SPI) and keys to apply to inbound and outbound protected traffic if the specified transform set includes the ESP protocol.
		Or
		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.

Step 9	**exit** **Example:** `Device(config-crypto-m)# exit`	Exits crypto map configuration mode and returns to global configuration mode.

	Command or Action	Purpose
Step 10	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode.
Step 11	**show crypto map** [**interface** *interface* | **tag** *map-name*] **Example:** `Device# show crypto map`	Displays your crypto map configuration.

- Troubleshooting Tips, page 40
- What to Do Next, page 40

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 entire 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 "Applying Crypto Map Sets to Interfaces, page 40" section.

Applying Crypto Map Sets to Interfaces

You must apply a crypto map set to each interface through which IPsec traffic flows. Applying the crypto map set to an interface instructs the device to evaluate 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 the crypto map.

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:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**interface** *type/number* **Example:** `Device(config)# interface FastEthernet 0/0`	Configures an interface and enters interface configuration mode.
Step 4	**crypto map** *map-name* **Example:** `Device(config-if)# crypto map mymap`	Applies a crypto map set to an interface.
Step 5	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 6	**crypto map** *map-name* **local-address** *interface-id* **Example:** `Device(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:** `Device(config)# exit`	(Optional) Exits global configuration mode.
Step 8	**show crypto map** [**interface** *interface*] **Example:** `Device# show crypto map`	(Optional) Displays your crypto map configuration

Configuration Examples for IPsec VPN

Example: Configuring AES-Based Static Crypto Map

This example shows how a static crypto map is configured and how an AES is defined as the encryption method:

```
crypto isakmp policy 10
 encryption aes 256
 authentication pre-share
 group 14
 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
 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
 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 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
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
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
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*
IKE configuration	*Configuring Internet Key Exchange for IPsec VPNs*
Suite-B SHA-2 family (HMAC variant) and Elliptic Curve (EC) key pair configuration	*Configuring Internet Key Exchange for IPsec VPNs*
Suite-B Integrity algorithm type transform configuration	*Configuring Internet Key Exchange Version 2 (IKEv2)*
Suite-B Elliptic Curve Digital Signature Algorithm (ECDSA) signature (ECDSA-sig) authentication method configuration for IKEv2	*Configuring Internet Key Exchange Version 2 (IKEv2)*
Suite-B Elliptic curve Diffie-Hellman (ECDH) support for IPsec SA negotiation	• *Configuring Internet Key Exchange for IPsec VPNs* • *Configuring Internet Key Exchange Version 2 (IKEv2) and FlexVPN Site-to-Site*
Suite-B support for certificate enrollment for a PKI	*Configuring Certificate Enrollment for a PKI*
Recommended cryptographic algorithms	*Next Generation Encryption*

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 software 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*
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 4 Feature Information for Configuring Security for IPsec VPNs

Feature Name	Software Releases	Feature Information
Advanced Encryption Standard	12.2(8)T	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 commands were modified by this feature: **crypto ipsec transform-set, encryption (IKE policy), show crypto ipsec transform-set, show crypto isakmp policy.**
DES/3DES/AES VPN Encryption Module (AIM-VPN/EPII, AIM-VPN/HPII, AIM-VPN/BPII Family)	12.3(7)T	This feature describes in which VPN encryption hardware AIM and NM are supported, in certain Cisco IOS software releases.
IKEv2 Proposal Support	15.1(1)T	An IKEv2 proposal is a set of transforms used in the negotiation of IKEv2 SA as part of the IKE_SA_INIT exchange. An IKEv2 proposal is regarded as complete only when it has at least an encryption algorithm, an integrity algorithm, and a Diffie-Hellman (DH) group configured. If no proposal is configured and attached to an IKEv2 policy, then the default proposal is used in negotiation. The following commands were modified by this feature: **crypto ikev2 proposal, encryption (ikev2 proposal), group (ikev2 proposal), integrity (ikev2 proposal), show crypto ikev2 proposal.**

Feature Name	Software Releases	Feature Information
IPv6 Support for IPsec and IKEv2	15.1(4)M 15.1(1)SY	This feature allows IPv6 addresses to be added to IPsec and IKEv2 protocols. The following commands were introduced or modified: **crypto map (global IPsec), crypto map (isakmp), crypto map (Xauth), ipv6 crypto map.**
Option to Disable Volume-based IPsec Lifetime Rekey	15.0(1)M	This feature allows customers to disable the IPsec security association rekey when processing large amounts of data. The following commands were modified by this feature: **crypto ipsec security association lifetime, set security-association lifetime.**
SEAL Encryption	12.3(7)T	This feature adds support for SEAL encryption in IPsec. The following command was modified by this feature: **crypto ipsec transform-set.**
Software IPPCP (LZS) with Hardware Encryption	12.2(13)T	This feature allows customers to use LZS software compression with IPsec when a VPN module is in Cisco 2600 and Cisco 3600 series routers.
Suite-B Support in IOS SW Crypto	15.1(2)T	Suite-B adds support for four user interface suites of cryptographic algorithms for use with IKE and IPSec that are described in RFC 4869. Each suite consists of an encryption algorithm, a digital signature algorithm, a key agreement algorithm, and a hash or message digest algorithm. The following command was modified by this feature: **crypto ipsec transform-set**.

IPsec Virtual Tunnel Interfaces

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 the configuration of IPsec for protection of remote links, support multicast, and simplify network management and load balancing.

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

Finding Feature Information

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

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 Interfaces

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. Therefore, 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.”

A dynamic VTI (DVTIs) is also a point-to-point interface that can support multiple IPsec SAs. The DVTI can accept the multiple IPsec selectors that are proposed by the initiator.

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.

DVTIs support multiple proxies, but DVTIs do not allow mixing “any any” proxies with non-“any any” proxies. DVTIs permit only one type of proxy at a time, either a single “any any” proxy or multiple “no any any” proxies.

Quality of Service (QoS) Traffic Shaping

The shaped traffic is process switched.

Stateful Failover

IPsec stateful failover is not supported with IPsec VTIs.

Static VTIs Versus GRE Tunnels

The IPsec VTI is limited to the IP unicast and multicast traffic only, as opposed to Generic Routing Encapsulation (GRE) tunnels, which have a wider application for IPsec implementation.

Single Template Model

In the single template model, the VPN routing and forwarding (VRF) is configured in the ISAKMP profile. In this model, each virtual access that is created belongs to the internal VRF (IVRF) specified in the ISAKMP profile. But because the IP address of the virtual access is derived from the interface to which the virtual access is unnumbered to, the IP address of the interface will not be available in the virtual access routing table. This happens because the unnumbered interface does not belong to the IVRF routing table of the virtual access. In such cases, a ping to the virtual access IP address fails.

Tunnel Protection

Do not configure the **shared** keyword when using the **tunnel mode ipsec ipv4** command for IPsec IPv4 mode.

Virtual Template Lock

Effective with CSCtt26236, the virtual template lock allows you to modify or delete a virtual template of type tunnel only when the virtual template is not associated with any cloned virtual access interfaces. The

virtual template lock prevents dynamic command updates from virtual templates to the cloned virtual access interfaces, which can cause instability in some scenarios.

If you try to modify or delete an active virtual template of type tunnel, the following error message appears:

```
Device(config)# interface virtual-template 1
 % Virtual-template config is locked, active vaccess present
```

Although the virtual template cannot be modified when the virtual template is associated with a virtual access interface, perform the following steps to modify an existing virtual template configuration:

1. Configure a new virtual template interface. For more information, see "Configuring Dynamic IPsec Virtual Tunnel Interfaces, page 62."
2. Associate the new virtual template to the IKEv2 profile. For more information, see the *Configuring IKEv2 Profile (Basic)* module.
3. Clear the active sessions using the **clear crypto session** command or wait for session termination.

The new session will use the new virtual template.

VRF-Aware IPsec Configuration

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

Information About IPsec Virtual Tunnel Interfaces

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 generic routing encapsulation (GRE) or Layer 2 Tunneling Protocol (L2TP) tunnels 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. Because DVTIs function like any other real interface you can apply quality of service (QoS), firewall, and other security services as soon as the tunnel is active.

Without 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 Network Address Translation (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 extra features to the IPsec tunnel.

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

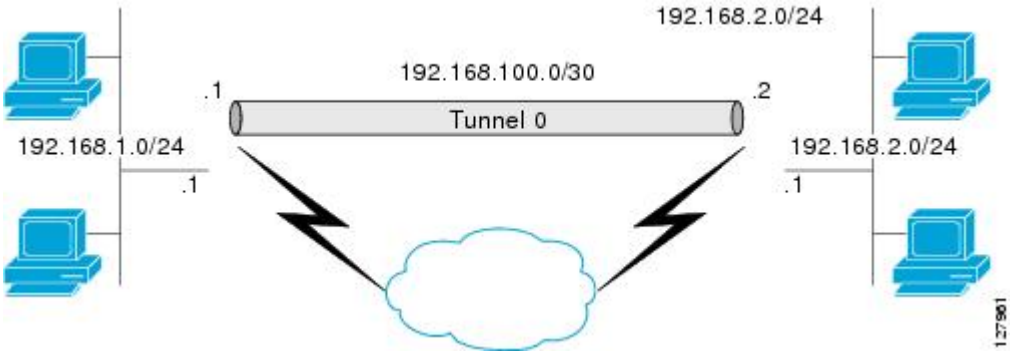
Static Virtual Tunnel Interfaces

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 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 5 IPsec SVTI



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

Note When configuring IPsec SVTI with high availability (HA), the standby router reload does not affect the existing security associations.

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.

DVTIs can be used for both the server and the 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 software feature configured on the virtual template interface, such as QoS, NetFlow, or ACLs.

DVTIs function like any other real interface, so you can apply QoS, firewall, or 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 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 an extended authentication (Xauth) User or Unity group, or 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 VPN routing and forwarding- (VRF-) aware IPsec deployment. The VRF is configured on the interface.

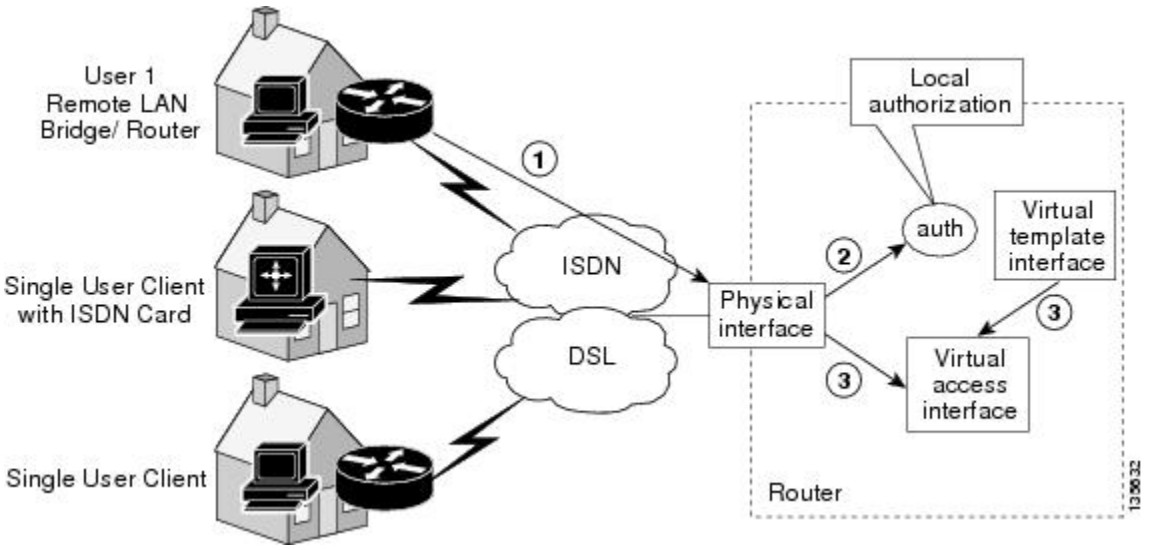
A DVTI requires minimal configuration on the router. A single virtual template can be configured and cloned.

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 only in Easy VPNs. That is, the DVTI end must be configured as an Easy VPN server.

The figure below illustrates the DVTI authentication path.

Figure 6 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.

Dynamic Virtual Tunnel Interface Life Cycle

IPsec profiles define the 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

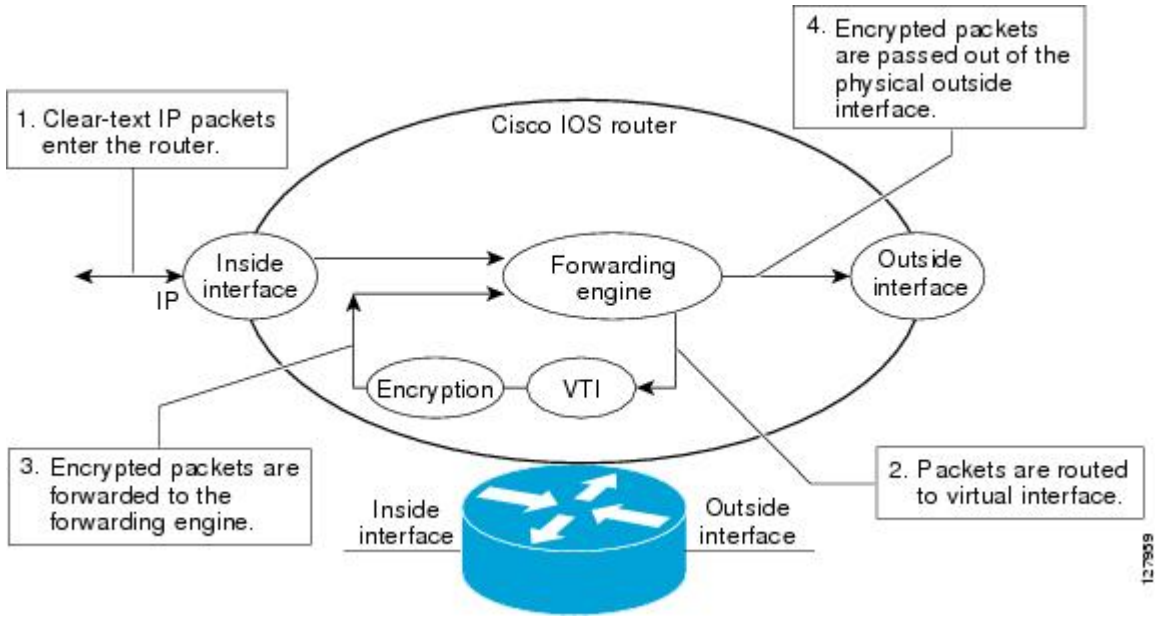
Because VTIs are routable interfaces, routing plays an important role in the encryption process. Traffic is encrypted only if it is forwarded out of the VTI, and traffic arriving on the VTI is decrypted and routed accordingly. VTIs allow you to establish an encryption tunnel using a real interface as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

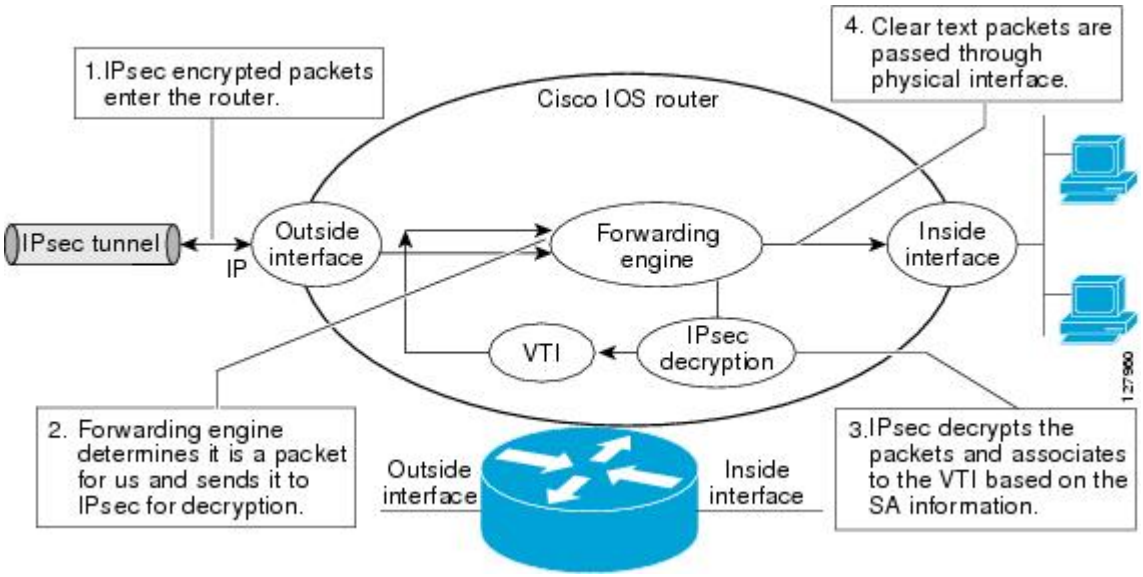
Figure 7 Packet Flow into the IPsec Tunnel



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



How to Configure IPsec Virtual Tunnel Interfaces

- Configuring Static IPsec Virtual Tunnel Interfaces, page 56
- Configuring User-Defined Static IPsec Virtual Tunnel Interfaces, page 58
- Configuring Dynamic IPsec Virtual Tunnel Interfaces, page 62

Configuring Static IPsec Virtual Tunnel Interfaces

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. **exit**
6. **interface** *type number*
7. **ip address** *address mask*
8. **tunnel mode ipsec ipv4**
9. **tunnel source** *interface-type interface-type*
10. **tunnel destination** *ip-address*
11. **tunnel protection IPsec profile** *profile-name* [**shared**]
12. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto IPsec profile** *profile-name* **Example:** `Device(config)# crypto IPsec profile PROF`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec devices, and enters IPsec profile configuration mode.
Step 4	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Device(ipsec-profile)# set transform-set tset`	Specifies which transform sets can be used with the crypto map entry.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits IPsec profile configuration mode, and enters global configuration mode.
Step 6	**interface** *type number* **Example:** `Device(config)# interface tunnel 0`	Specifies the interface on which the tunnel will be configured and enters interface configuration mode.
Step 7	**ip address** *address mask* **Example:** `Device(config-if)# ip address 10.1.1.1 255.255.255.0`	Specifies the IP address and mask.
Step 8	**tunnel mode ipsec ipv4** **Example:** `Device(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 9	**tunnel source** *interface-type interface-type* **Example:** `Device(config-if)# tunnel source loopback 0`	Specifies the tunnel source as a loopback interface.
Step 10	**tunnel destination** *ip-address* **Example:** `Device(config-if)# tunnel destination 172.16.1.1`	Identifies the IP address of the tunnel destination.
Step 11	**tunnel protection IPsec profile** *profile-name* [**shared**] **Example:** `Device(config-if)# tunnel protection IPsec profile PROF`	Associates a tunnel interface with an IPsec profile.

	Command or Action	Purpose
Step 12	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring User-Defined Static IPsec Virtual Tunnel Interfaces

SUMMARY STEPS

1. enable
2. configure terminal
3. crypto keyring *keyring name*
4. crypto keyring vrf *vrf name*
5. pre-shared-key address *address*
6. pre-shared-key address key *key*
7. exit
8. interface *type number*
9. ip vrf forwarding *vrf name*
10. ip address *address mask*
11. tunnel source *address*
12. tunnel destination *address*
13. tunnel mode ipsec ipv4
14. tunnel vrf *vrf name*
15. tunnel protection IPsec profile *profile name*
16. exit
17. interface *interface-type*
18. ip vrf forwarding *vrf name*
19. ip address *ip-address*
20. no shutdown
21. end

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	configure terminal **Example:** `Device# configure terminal`	Enables global configuration mode.
Step 3	crypto keyring *keyring name* **Example:** `Device(config)# crypto keyring KR-FVRF`	Defines the crypto keyring name to which the keyring will be referenced.
Step 4	crypto keyring vrf *vrf name* **Example:** `Device(config)# crypto keyring vrf FVRF`	Defines the crypto keyring name for a virtual routing and forwarding (VRF) instance to which the keyring will be referenced.
Step 5	pre-shared-key address *address* **Example:** `Device(config-keyring)# pre-shared-key address 10.0.0.2`	Defines the preshared key IP address of the remote host.
Step 6	pre-shared-key address key *key* **Example:** `Device(config-keyring)# pre-shared-key address 10.0.0.2 key cisco`	Defines the secret key for the preshared key IP address of the remote host.
Step 7	exit **Example:** `Device(config-keyring)# exit`	Exits keyring configuration mode, and enters global configuration mode.
Step 8	interface *type number* **Example:** `Device(config)# interface tunnel 0`	Specifies the interface on which the tunnel will be configured and enters interface configuration mode.

	Command or Action	Purpose
Step 9	ip vrf forwarding *vrf name* **Example:** `Device(config-if)# ip vrf forwarding IVRF`	Specifies the virtual routing and forwarding (VRF) name for the interface.
Step 10	ip address *address mask* **Example:** `Device(config-if)# ip address 10.0.0.1 255.255.255.0`	Specifies the IP address and address mask.
Step 11	tunnel source *address* **Example:** `Device(config-if)# tunnel source 10.0.0.1`	Specifies the IP address of the tunnel source.
Step 12	tunnel destination *address* **Example:** `Device(config-if)# tunnel destination 10.0.0.2`	Specifies the IP address of the tunnel destination.
Step 13	tunnel mode ipsec ipv4 **Example:** `Device(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 14	tunnel vrf *vrf name* **Example:** `Device(config-if)# tunnel vrf FVRF`	Defines the virtual routing and forwarding (VRF) instance for the tunnel.
Step 15	tunnel protection IPsec profile *profile name* **Example:** `Device(config-if)# tunnel protection IPsec profile IPSec-Profile`	Associates a tunnel interface with an IPsec profile.

	Command or Action	Purpose
Step 16	exit **Example:** `Device(config-if)# exit`	Exits interface configuration mode, and enters global configuration mode.
Step 17	interface *interface-type* **Example:** `Device(config)# interface FastEthernet0/0`	Specifies the interface type.
Step 18	ip vrf forwarding *vrf name* **Example:** `Device(config-if)# ip vrf forwarding FVRF`	Specifies the VRF forwarding table for the interface.
Step 19	ip address *ip-address* **Example:** `Device(config-if)# ip address 10.0.0.1 255.255.255.0`	Specifies the IP address and address mask.
Step 20	no shutdown **Example:** `Device(config-if)# no shutdown`	Enables the interface (brings it up).
Step 21	end **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring Dynamic IPsec Virtual Tunnel Interfaces

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. **exit**
6. **interface virtual-template** *number*
7. **tunnel mode ipsec ipv4**
8. **tunnel protection IPsec profile** *profile-name* [**shared**]
9. **exit**
10. **crypto isakamp profile** *profile-name*
11. **match identity address** *ip-addressmask*
12. **virtual template** *template-number*
13. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec profile** *profile-name* **Example:** `Device(config)# crypto ipsec profile PROF`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec devices and enters IPsec profile configuration mode.

	Command or Action	Purpose
Step 4	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Device(ipsec-profile)# set transform-set tset`	Specifies which transform sets can be used with the crypto map entry.
Step 5	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits ipsec profile configuration mode and enters global configuration mode.
Step 6	**interface virtual-template** *number* **Example:** `Device(config)# interface virtual-template 2`	Defines a virtual-template tunnel interface and enters interface configuration mode.
Step 7	**tunnel mode ipsec ipv4** **Example:** `Device(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 8	**tunnel protection IPsec profile** *profile-name* [**shared**] **Example:** `Device(config-if)# tunnel protection ipsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 9	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 10	**crypto isakamp profile** *profile-name* **Example:** `Device(config)# crypto isakamp profile profile1`	Defines the ISAKAMP profile to be used for the virtual template.

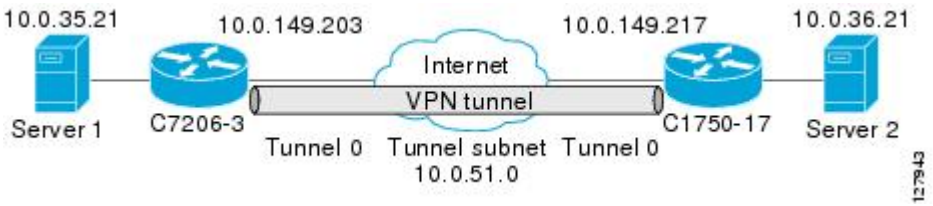
	Command or Action	Purpose
Step 11	**match identity address** *ip-addressmask* **Example:** `Device(conf-isa-prof)# match identity address 10.1.1.0 255.255.255.0`	Matches an identity from the ISAKMP profile and enters isakmp-profile configuration mode.
Step 12	**virtual template** *template-number* **Example:** `Device(config)# virtual-template 1`	Specifies the virtual template attached to the ISAKAMP profile.
Step 13	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuration Examples for IPsec Virtual Tunnel Interfaces

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 the 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



Cisco 7206 Router Configuration

```
version 12.3
service timestamps debug datetime
service timestamps log datetime
hostname 7200-3
no aaa new-model
ip subnet-zero
ip cef
controller ISA 6/1
!
crypto isakmp policy 1
encr aes
authentication pre-share
group 14
crypto isakmp key Cisco12345 address 0.0.0.0 0.0.0.0
crypto ipsec transform-set T1 esp-aes 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 Ethernet3/0
 ip address 10.0.149.203 255.255.255.0
 duplex full
!
interface Ethernet3/3
 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
```

Cisco 1750 Router Configuration

```
version 12.3
hostname c1750-17
no aaa new-model
ip subnet-zero
ip cef
crypto isakmp policy 1
encr aes
authentication pre-share
group 14
crypto isakmp key Cisco12345 address 0.0.0.0 0.0.0.0
crypto ipsec transform-set T1 esp-aes 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 Ethernet1/0
 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 66

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 Cisco 7206 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, Ethernet3/3
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, Ethernet3/0
```

Example: VRF-Aware Static Virtual Tunnel Interface

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

Cisco 7206 Router Configuration

```
hostname cisco 7206
.
.
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 shows how to police traffic out the tunnel interface.

Cisco 7206 Router Configuration

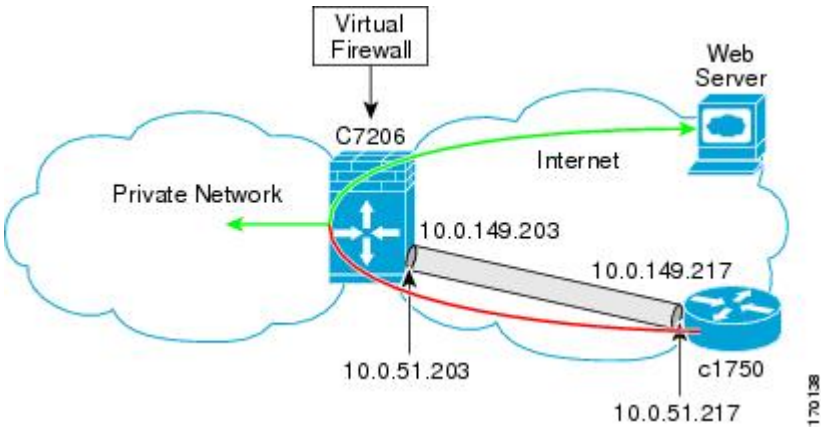
```
hostname cisco 7206
.
.
class-map match-all VTI
 match any
!
policy-map VTI
  class VTI
  police cir 2000000
    conform-action transmit
    exceed-action drop
!
.
.
interface Tunnel0
 ip address 10.0.51.217 255.255.255.0
 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
```

```
 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 an 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:

Cisco 7206 Router Configuration

```
hostname cisco 7206
.
.
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 a Cisco IOS router configured as an Easy VPN client.

Cisco 7206 Router Configuration

```
hostname cisco 7206
!
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 aes
 authentication pre-share
 group 14
!
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-aes 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 70

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 the same idea as the Easy VPN client that you can run from a PC to connect to a network. The configuration of the Easy VPN server will work for the software client or the Cisco IOS client.

```
hostname cisco 1841
!
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, the IPsec tunnel has to be initiated manually by a user.

Note the use of the **mode** command. The mode can be a client, network-extension, or network-extension-plus. This example indicates the client mode, which means that the client is given a private address from the server. The network-extension mode is different from the client mode in that the client specifies for the server its attached private subnet. Depending on the mode, the routing table on either end will be 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 71

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

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

```
hostname c7206
.
.
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: Dynamic Virtual Tunnel Interface with 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 cisco 7206
.
.
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 will also be applied to the virtual access interface. The following example shows the basic DVTI configuration with QoS added.

```
hostname cisco 7206
.
.
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
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*
Security commands	• *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*
IPsec configuration	*Configuring Security for VPNs with IPsec*
QoS configuration	*Cisco IOS Quality of Service Solutions Configuration Guide*
VPN configuration	• *Cisco Easy VPN Remote* • *Easy VPN Server*
Recommended cryptographic algorithms	*Next Generation Encryption*

Standards and RFCs

Standard/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 Interfaces

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 5 Feature Information for IPsec Virtual Tunnel Interfaces

Feature Name	Releases	Feature Configuration Information
Dynamic IPsec VTIs	12.3(7)T 12.3(14)T	Dynamic VTIs enable efficient 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. IPsec dynamic VTIs allow you to create highly secure connectivity for remote access VPNs. The dynamic VTI simplifies VRF-aware IPsec deployment. The following commands were introduced or modified: **crypto isakmp profile, interface virtual-template, show vtemplate, tunnel mode, virtual-template.**

Feature Name	Releases	Feature Configuration Information
Multi-SA for Dynamic VTIs	15.2(1)T	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.**
Static IPsec VTIs	12.2(33)SRA 12.2(33)SXH 12.3(7)T 12.3(14)T	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.

SafeNet IPsec VPN Client Support

The SafeNet IPsec VPN Client Support feature allows you to limit the scope of an Internet Security Association and Key Management Protocol (ISAKMP) profile or ISAKMP keyring configuration to a local termination address or interface. The benefit of this feature is that different customers can use the same peer identities and ISAKMP keys by using different local termination addresses.

History for the SafeNet IPsec VPN Client Support Feature

Release	Modification
12.3(14)T	This feature was introduced.
12.2(18)SXE	This feature was integrated into Cisco IOS Release 12.2(18)SXE.

Finding Feature Information

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

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 SafeNet IPsec VPN Client Support

- You must understand how to configure ISAKMP profiles and ISAKMP keyrings.

Restrictions for SafeNet IPsec VPN Client Support

- The local address option works only for the primary address of an interface.
- If an IP address is provided, the administrator has to ensure that the connection of the peer terminates to the address that is provided.
- If the IP address does not exist on the device, or if the interface does not have an IP address, the ISAKMP profile or ISAKMP keyring will be effectively disabled.

Information About SafeNet IPsec VPN Client Support

ISAKMP Profile and ISAKMP Keyring Configurations Background

Prior to Cisco IOS Release 12.3(14)T, ISAKMP-profile and ISAKMP-keyring configurations could be only global, meaning that the scope of these configurations could not be limited by any locally defined parameters (VRF instances were an exception). For example, if an ISAKMP keyring contained a preshared key for address 10.11.12.13, the same key would be used if the peer had the address 10.11.12.13, irrespective of the interface or local address to which the peer was connected. There are situations, however, in which users prefer that associate keyrings be bound not only with virtual route forwarding (VRF) instances but also to a particular interface. For example, if instead of VRF instances, there are virtual LANS, and the Internet Key Exchange (IKE) is negotiated with a group of peers using one fixed virtual LAN (VLAN) interface. Such a group of peers uses a single preshared key, so if keyrings could be bound to an interface, it would be easy to define a wildcard key without risking that the keys would also be used for other customers.

Sometimes the identities of the peer are not in the control of the administrator, and even if the same peer negotiates for different customers, the local termination address is the only way to distinguish the peer. After such a distinction is made, if the traffic is sent to different VRF instances, configuring an ISAKMP profile is the only way to distinguish the peer. Unfortunately, when the peer uses an identical identity for all such situations, the ISAKMP profile cannot distinguish among the negotiations. For such scenarios, it would be beneficial to bind ISAKMP profiles to a local termination address. If a local termination address could be assigned, identical identities from the peer would not be a problem.

Local Termination Address or Interface

Effective with Cisco IOS Release 12.3(14)T, the SafeNet IPsec VPN Client Support feature allows you to limit the scope of ISAKMP profiles and ISAKMP keyrings to a local termination address or interface.

Benefit of SafeNet IPsec VPN Client Support

The benefit of this feature is that different customers can use the same peer identities and ISAKMP keys by using different local termination addresses.

How to Configure SafeNet IPsec VPN Client Support

This section contains the following procedures. The first two configurations are independent of each other.

Limiting an ISAKMP Profile to a Local Termination Address or Interface

To configure an ISAKMP profile and limit it to a local termination address or interface, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto isakmp profile** *profile-name*
4. **keyring** *keyring-name*
5. **match identity address** *address*
6. **local-address** {*interface-name* | *ip-address* [*vrf-tag*]}

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 isakmp profile** *profile-name* **Example:** `Router (config)# crypto isakmp profile profile1`	Defines an ISAKMP profile and enters ISAKMP profile configuration mode.

	Command or Action	Purpose
Step 4	**keyring** *keyring-name* **Example:** `Router (conf-isa-profile)# keyring keyring1`	(Optional) Configures a keyring with an ISAKMP profile. • A keyring is not needed inside an ISAKMP profile for local termination to work. Local termination works even if Rivest, Shamir, and Adelman (RSA) certificates are used.
Step 5	**match identity address** *address* **Example:** `Router (conf-isa-profile)# match identity address 10.0.0.0 255.0.0.0`	Matches an identity from a peer in an ISAKMP profile.
Step 6	**local-address** {*interface-name* | *ip-address* [*vrf-tag*]} **Example:** `Router (conf-isa-profile)# local-address serial2/0`	Limits the scope of an ISAKMP profile or an ISAKMP keyring configuration to a local termination address or interface.

Limiting a Keyring to a Local Termination Address or Interface

To configure an ISAKMP keyring and limit its scope to a local termination address or interface, perform the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto keyring** *keyring-name*
4. **local-address** {*interface-name* |*ip-address*[*vrf-tag*]}
5. **pre-shared-key address** *address*

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 keyring** *keyring-name* **Example:** `Router (config)# crypto keyring keyring1`	Defines a crypto keyring to be used during IKE authentication and enters keyring configuration mode.
Step 4	**local-address** {*interface-name* |*ip-address*[*vrf-tag*]} **Example:** `Router (conf-keyring)# local-address serial2/0`	Limits the scope of an ISAKMP profile or an ISAKMP keyring configuration to a local termination address or interface.
Step 5	**pre-shared-key address** *address* **Example:** `Router (conf-keyring)# pre-shared-key address 10.0.0.1`	Defines a preshared key to be used for IKE authentication.

Monitoring and Maintaining SafeNet IPsec VPN Client Support

The following **debug** and **show** commands may be used to monitor and maintain the configuration in which you limited the scope of an ISAKMP profile or ISAKMP keyring to a local termination address or interface.

SUMMARY STEPS

1. **enable**
2. **debug crypto isakmp**
3. **show crypto isakmp profile**

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** **Example:** `Router# debug crypto isakmp`	Displays messages about IKE events.
Step 3	**show crypto isakmp profile** **Example:** `Router# show crypto isakmp profile`	Lists all the ISAKMP profiles that are defined on a router.

- Examples, page 82

Examples

- debug crypto isakmp Command Output for an ISAKMP Keyring That IsBound to Local Termination Addresses Example, page 82
- debug crypto isakmp Command Output for an ISAKMP ProfileThat Is Boundto a Local Termination Address Example, page 83
- show crypto isakmp profile Command Output Example, page 83

debug crypto isakmp Command Output for an ISAKMP Keyring That IsBound to Local Termination Addresses Example

You have an ISAKMP configuration as follows (the address of serial2/0 is 10.0.0.1, and the address of serial2/1 is 10.0.0.2),

```
crypto keyring keyring1
! Scope of the keyring is limited to interface serial2/0.
  local-address serial2/0
  ! The following is the key string used by the peer.
  pre-shared-key address 10.0.0.3 key somerandomkeystring
crypto keyring keyring2
  local-address serial2/1
  ! The following is the keystring used by the peer coming into serial2/1.
   pre-shared-key address 10.0.0.3 key someotherkeystring
```

and if the connection is coming into serial2/0, keyring1 is chosen as the source of the preshared key (and keyring2 is ignored because it is bound to serial2/1), you would see the following output:

```
Router# debug crypto isakmp
*Feb 11 15:01:29.595: ISAKMP:(0:0:N/A:0):Keyring keyring2 is bound to
```

```
  10.0.0.0, skipping
*Feb 11 15:01:29.595: ISAKMP:(0:0:N/A:0):Looking for a matching key for
  10.0.0.3 in keyring1
*Feb 11 15:01:29.595: ISAKMP:(0:0:N/A:0): : success
*Feb 11 15:01:29.595: ISAKMP:(0:0:N/A:0):found peer pre-shared key
  matching 10.0.0.3
*Feb 11 15:01:29.595: ISAKMP:(0:0:N/A:0): local preshared key found
```

debug crypto isakmp Command Output for an ISAKMP ProfileThat Is Boundto a Local Termination Address Example

If you have the following configuration,

```
crypto isakmp profile profile1
  keyring keyring1
  match identity address 10.0.0.0 255.0.0.0
  local-address serial2/0
crypto isakmp profile profile2
  keyring keyring1
  keyring keyring2
  self-identity fqdn
  match identity address 10.0.0.1 255.255.255.255
  local-address serial2/1
```

and the connection is coming through the local terminal address serial2/0, you will see the following output:

```
Router# debug crypto isakmp
*Feb 11 15:01:29.935: ISAKMP:(0:0:N/A:0):
Profile profile2 bound to 10.0.0.0 skipped
*Feb 11 15:01:29.935: ISAKMP:(0:1:SW:1):: peer matches profile1 profile
```

show crypto isakmp profile Command Output Example

The following is an example of typical **show** command output for an ISAKMP profile that is bound to serial2/0:

```
Router# show crypto isakmp profile
ISAKMP PROFILE profile1
  Identities matched are:
    ip-address 10.0.0.0 255.0.0.0
  Certificate maps matched are:
  keyring(s): keyring1
  trustpoint(s): <all>
  Interface binding: serial2/0 (10.20.0.1:global)
```

Troubleshooting SafeNet IPsec VPN Client Support

If an ISAKMP profile or ISAKMP keyring fails to be selected, you should double-check the local-address binding in the ISAKMP profile or ISAKMP keyring configuration and follow the output of the IKE debugs to determine whether the peer is correctly terminating on the address. You may remove the local-address binding (to make the scope of the profile or keyring global) and check to determine whether the profile or keyring is selected to confirm the situation.

Configuration Examples for SafeNet IPsec VPN Client Support

This section contains the following configuration, **debug** command, and **show** command examples.

ISAKMP Profile Bound to a Local Interface Example

The following example shows that the ISAKMP profile is bound to a local interface:

```
crypto isakmp profile profile1
  keyring keyring1
  match identity address 10.0.0.0 255.0.0.0
```

local-address serial2/0

ISAKMP Keyring Bound to a Local Interface Example

The following example shows that the ISAKMP keyring is bound only to interface serial2/0:

```
crypto keyring
  local-address serial2/0
```

pre-shared-key address 10.0.0.1

ISAKMP Keyring Bound to a Local IP Address Example

The following example shows that the ISAKMP keyring is bound only to IP address 10.0.0.2:

```
crypto keyring keyring1
  local-address 10.0.0.2
```

pre-shared-key address 10.0.0.2 key

ISAKMP Keyring Bound to an IP Address and Limited to a VRF Example

The following example shows that an ISAKMP keyring is bound to IP address 10.34.35.36 and that the scope is limited to VRF examplevrf1:

```
ip vrf examplevrf1
  rd 12:3456
crypto keyring ring1
  local-address 10.34.35.36 examplevrf1
interface ethernet2/0
  ip vrf forwarding examplevrf1
  ip address 10.34.35.36 255.255.0.0
```

Additional References

The following sections provide references related to SafeNet IPsec VPN Client Support.

Related DocumentsStandards

Related Topic	Document Title
Configuring ISAKMP profiles and ISAKMP keyrings	VRF-Aware IPsec
Security commands	*Cisco IOS Security Command Reference*

Standard	Title
No new or modified standards are supported by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported 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
No new or modified RFCs are supported by this feature.	--

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

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.

Feature History for Crypto Conditional Debug Support

Feature History	
Release	Modification
12.3(2)T	This feature was introduced.

Finding Feature Information

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

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 6 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).

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
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.
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

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.

	Command or Action	Purpose
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.
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

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" section of *Cisco IOS 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 software 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
Technical Assistance Center (TAC) home page, containing 30,000 pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport