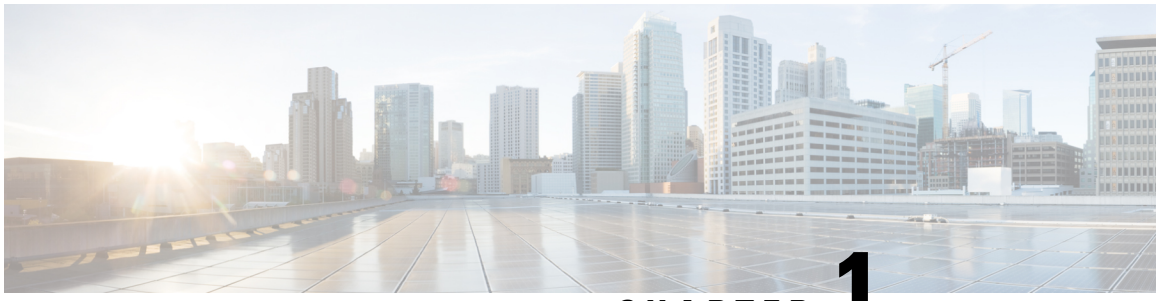
Security for VPNs with IPsec Configuration Guide, Cisco IOS XE Gibraltar 16.12.x

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



CHAPTER 1

Read Me First

Important Information

Note For CUBE feature support information in Cisco IOS XE Bengaluru 17.6.1a and later releases, see Cisco Unified Border Element IOS-XE Configuration Guide.

Note The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.

Feature Information

Use Cisco Feature Navigator to find information about feature support, platform support, and Cisco software image support. An account on Cisco.com is not required.

Related References

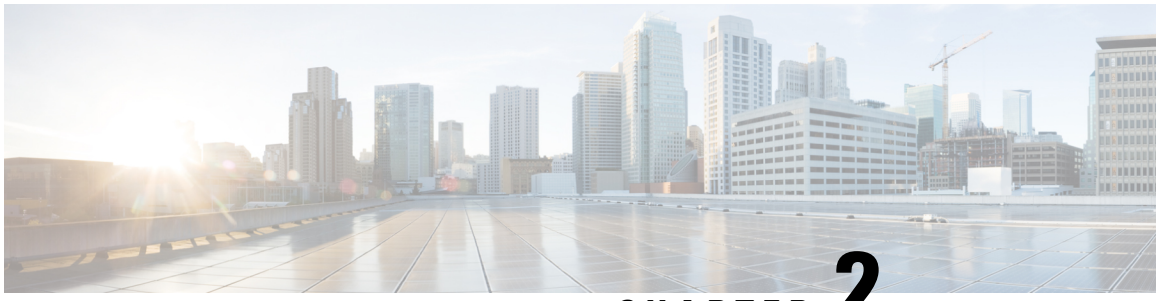
- Cisco IOS Command References, All Releases

Obtaining Documentation and Submitting a Service Request

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Short Description

The documentation set for this product strives to use bias-free language. For purposes of this documentation set, bias-free is defined as language that does not imply discrimination based on age, disability, gender, racial identity, ethnic identity, sexual orientation, socioeconomic status, and intersectionality. Exceptions may be present in the documentation due to language that is hardcoded in the user interfaces of the product software, language used based on standards documentation, or language that is used by a referenced third-party product.



CHAPTER 2

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.

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

Cisco IPsec Policy Map MIB

The MIB OID objects are displayed only when an IPsec session is up.

Discontiguous Access Control Lists

Crypto maps using access control lists (ACLs) that have discontiguous masks are not supported.

Physical Interface and Crypto Map

A crypto map on a physical interface is not supported, if the physical interface is the source interface of a tunnel protection interface.

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.

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.

Unsupported Interface Types

- Crypto VPNs are not supported on the bridge domain interfaces (BDI).

- Crypto maps are not supported on tunnel interface and port-channel interface. As an exception, crypto maps for GDOI are supported on tunnel interfaces.
- Crypto maps are not supported on loopback interfaces.
- If transport profile is enabled on a tunnel, crypto maps are not supported on the tunnel source interfaces.
- Crypto maps are not supported on tunnel interface of MFR.
- Crypto maps are not supported on Vlan interfaces
- GetVPN crypto map is supported on port-channel interfaces.

Information About Configuring Security for VPNs with IPsec

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 (IKEv1 and IKEv2)—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.

Note Starting from Cisco IOS XE Bengaluru 17.6.x, configuring a weak crypto algorithm generates a warning, but the warning can be safely ignored and does not impact the working of the algorithms. The following example displays a warning message for a weak crypto algorithm:

```
Device(config-ikev2-proposal)# group 5
%Warning: weaker dh-group is deprecated
```

The following table lists all the weak algorithms.

IKEv1	IKEv2	IPsec
DH_GROUP_768_MODP/Group 1	DH_GROUP_768_MODP/Group 1	ah-md5-hmac

IKEv1	IKEv2	IPsec
DH_GROUP_1024_MODP/Group 2	DH_GROUP_1024_MODP/Group 2	ah-sha-hmac
DH_GROUP_1536_MODP/Group 5	DH_GROUP_1536_MODP/Group 5	esp-des
DES	DES	esp-3des
3DES	3DES	esp-sha-hmac
MD5	MD5	esp-gmac
		esp-md5-hmac
		esp-null

The component technologies implemented for IPsec include:

- 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.

- 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.

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.

Note Starting from Cisco IOS XE Bengaluru 17.6.x, if the ISAKMP policy is enabled, the default algorithms that are available for configuration are:

- Encryption: AES
- Hash: SHA
- DH Group: 14

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 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, 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.

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).

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.

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.

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 sha1 md5
group 5 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.

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.

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.)

The table below shows allowed transform combinations.

Table 1: 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 recommended).
	ah-sha-hmac	AH with the SHA (Secure Hash Algorithm) (an HMAC variant) authentication algorithm.

Transform Type	Transform	Description	
ESP Encryption Transform (*Pick only one.*)	**esp-aes**	ESP with the 128-bit Advanced Encryption Standard (AES) encryption algorithm.	
	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 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.	

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.

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

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 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.

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.

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

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" 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" and "Applying Crypto Map Sets to Interfaces" 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.

	Command or Action	Purpose
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" 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.
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

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

	Command or Action	Purpose
		• 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.
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

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

```
crypto ikev2 proposal proposal-2
encryption aes-cbc-128 aes-cbc-192
integrity sha1 sha256
group 14 15
```

For a list of transform combinations, see Configuring Security for VPNs with IPsec.

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* 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. **crypto ipsec security-association dummy** {**pps** *rate* | **seconds** *seconds*}
7. **set transform-set** *transform-set-name1* [*transform-set-name2...transform-set-name6*]
8. **set security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes* | **kilobytes disable**}
9. **set security-association level per-host**
10. **set pfs** [**group1** | **group14** | **group15** | **group16** | **group19** | **group2** | **group20** | **group24 | group5**]
11. **end**
12. **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.
Step 6	**crypto ipsec security-association dummy** {**pps** *rate*	**seconds** *seconds*} **Example:** `Device(config-crypto-m)# set security-association dummy seconds 5`	Enables generating dummy packets. These dummy packets are generated for all flows created in the crypto map.

	Command or Action	Purpose
Step 7	**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 8	**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 9	**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 10	**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 11	**end** **Example:** `Device(config-crypto-m)# end`	Exits crypto map configuration mode and returns to privileged EXEC mode.
Step 12	**show crypto map** [**interface** *interface* | **tag** *map-name*] **Example:** `Device# show crypto map`	Displays your crypto map configuration.

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" 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.
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.

	Command or Action	Purpose
		• 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.
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.

	Command or Action	Purpose
		• 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.
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

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" 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.

	Command or Action	Purpose
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.
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:**	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.

	Command or Action	Purpose
	`Device(config-crypto-m)# set session-key outbound esp 256 cipher abcdefabcdefabcd`	
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 map** [**interface** *interface* | **tag** *map-name*] **Example:** `Device# show crypto map`	Displays your crypto map configuration.

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" 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 for Configuring Security for VPNs with IPsec

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
IKE, IPsec, and PKI configuration 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
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 Configuring 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 2: Feature Information for Configuring Security for IPsec VPNs

Feature Name	Software Releases	Feature Information
Advanced Encryption Standard		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**.
Suite-B Support in IOS SW Crypto		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**.

Note GetVPN crypto map is supported on port-channel interfaces from IOS XE 16.9.1 onwards.

Glossary

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

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

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

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

IKE—Internet Key Exchange. IKE establishes a shared security policy and authenticates keys for services (such as IPSec) that require keys. Before any IPSec traffic can be passed, each router/firewall/host must verify the identity of its peer. This can be done by manually entering preshared keys into both hosts or by a CA service.

IPsec—IP Security. 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 the negotiation of protocols and algorithms based on local policy and to generate the encryption and authentication keys to be used by IPSec. IPSec can 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.

peer—In the context of this module, a "peer" is a router or other device that participates in IPsec.

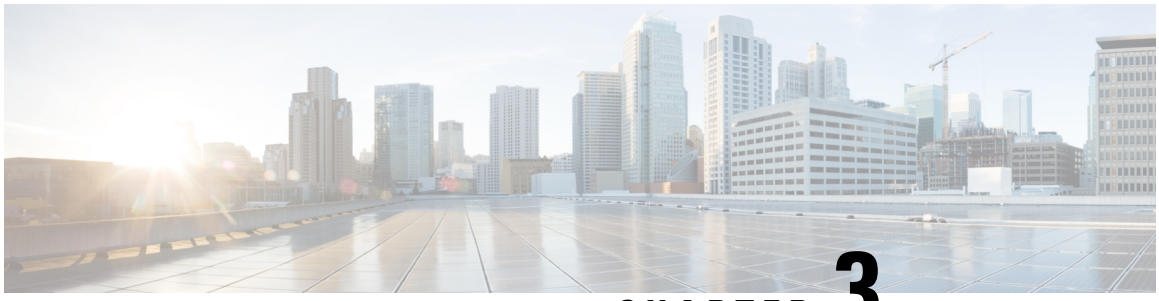
PFS—perfect forward secrecy. Cryptographic characteristic associated with a derived shared secret value. With PFS, if one key is compromised, previous and subsequent keys are not compromised, because subsequent keys are not derived from previous keys.

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

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

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

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



CHAPTER 3

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.

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

Fragmentation

Fragmentation is not supported over IPsec tunnel. You can choose to set the lower MTU on hosts to avoid packet fragments or choose to fragment the packets on any device.

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.

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.”

By default, Static VTIs (SVTIs) support only a single IPSec SA that is attached to the virtual tunnel interface. The traffic selector for the IPsec SA is always “IP any any”.

IPv4

This feature supports SVTIs that are configured to encapsulate IPv4 packets .

Tunnel Protection

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

Traceroute

The traceroute function with crypto offload on VTIs is not supported.

VxLAN GPE Tunnel Interface

The VxLAN GPE Tunnel Interface cannot use the same source interface as IPsec VTI.

Information About IPsec Virtual Tunnel Interfaces

The use of IPsec VTIs can simplify the configuration process when you need to provide protection for remote access and it provides an alternative to using generic routing encapsulation (GRE) or Layer 2 Tunneling Protocol (L2TP) tunnels for encapsulation. A benefit of using IPsec VTIs is that the configuration does not require 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 . 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.

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.

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

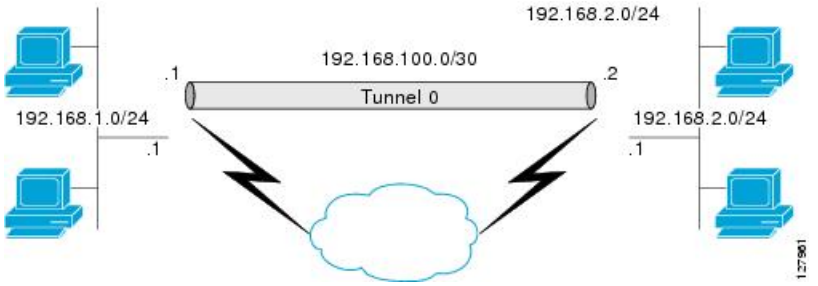
Static Virtual Tunnel Interfaces

SVTI configurations can be used for site-to-site connectivity in which a tunnel provides always-on access between two sites.

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 1: IPsec SVTI



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

Multi-SA Support for SVTI

By default, the traffic selector for an SVTI is set to ‘any any’. As a result, a single IPSec SA is attached for the SVTI corresponding to the ‘any any’ traffic selector.

From Cisco IOS XE Gibraltar 16.12.1, you can define and associate an Access Control List (ACL) with an SVTI to select traffic between specific source and destination proxies instead of the ‘any any’ proxy defined by the default. IPSec SAs are created for each non-any-any traffic selector, and thus, multiple SAs are attached to an SVTI.

This feature is supports IPv4 and IPv6 traffic protection with IPSec encapsulation in tunnel mode. The feature supports both IKEv1 and IKEv2.

Restrictions

- This feature is not supported with tunnel protection shared.
- This feature is not supported with IPSec Mixed Mode.
- Traffic selectors associated with the SVTIs at both the ends of a tunnel must have matching source and destination proxies. Do not narrow down the traffic selector at one of the SVTIs forming a tunnel.

ACL Characteristics and Effects on SVTI IPSec SAs

- An ACL associated with an SVTI must not contain an 'any any' proxy. For an 'any any' traffic selector, use the default behaviour of the SVTI and do not associate an ACL with the SVTI.
- An ACL associated with an SVTI supports only **permit** statements and must not contain **deny** statements.
- Run-time modification of an ACL associated with an SVTI is not supported. Shut the tunnel down before adding or modifying ACEs in the ACL.
- If you disassociate an ACL from an SVTI, existing IPSec SAs are deleted and a new IPSec SA for default traffic selector of 'IP any any' is formed.
- We recommend that you associate a maximum of 100 Access Control Entries (ACEs) with an SVTI. Further, all the ACLs associated with the various tunnel interfaces should together use a maximum of 2000 ACEs.

Reverse Route Injection

For Multi-SA SVTIs, Reverse Route Injection (RRI) can be configured in the IPSec profile.

If you use extended ACL or ACE options, such as protocol, port number, and DHCP, do not use RRI; use other means such as route maps for routing.

Note RRI capability with distance and tag is yet to be supported.

Dual Stack Support for SVTI

SVTI Dual Stack feature provides the capabilities to carry both IPv4 and IPv6 traffic using a single IPsec Security Association (SA) that is tunnelled over IPv4. From IOS XE release 17.9 onwards, Cisco supports specific subnets in ACL when the ingress end of the tunnel interface is configured with a third party IPSec client. Also, based on the third party IPsec client configuration, it responds with a specific traffic selector. In this case, the IPsec supports non-any non-any proxy configuration and allows to carry IPv4 or IPv6 type of traffic in the tunnel interface. This feature is supported only with IKEv2.

Restrictions

- Tunnel-mode configuration is allowed only under the IPsec profiles when you use the tunnel interface in dual-overlay mode.
- In Cisco IOS XE, ACL filtering infrastructure does not work on traffic generated locally on the device.
- You have to use the same set of traffic selectors for rekeying an IPsec SA. You cannot change the traffic selectors during the rekey process but when you change, the rekey request is rejected with the message *TS_UNACCEPTABLE*.
- A maximum of 16 traffic selectors are accepted at the IKEv2 level.
- ACLs on dual-stack tunnel interface are not supported. Any ACL configured on this interface is overwritten by dual-stack ACLs.

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.

Note You can configure DVTIs with IKEv1 or IKEv2. The legacy crypto map based configuration supports DVTIs with IKEv1 only. A DVTI configuration with IKEv2 is supported only in FlexVPN.

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.

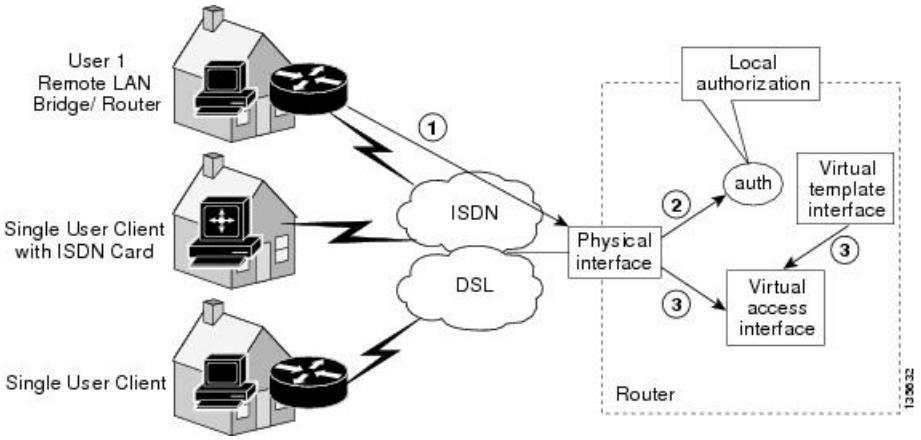
DVTIs provide efficiency in the use of IP addresses and provide secure connectivity. DVTIs allow dynamically downloadable per-group and per-user policies to be configured on a RADIUS server. The per-group or per-user definition can be created using an extended authentication (Xauth) User or Unity group, 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.

The figure below illustrates the DVTI authentication path.

Figure 2: Dynamic IPsec VTI



The authentication shown in the figure above follows this path:

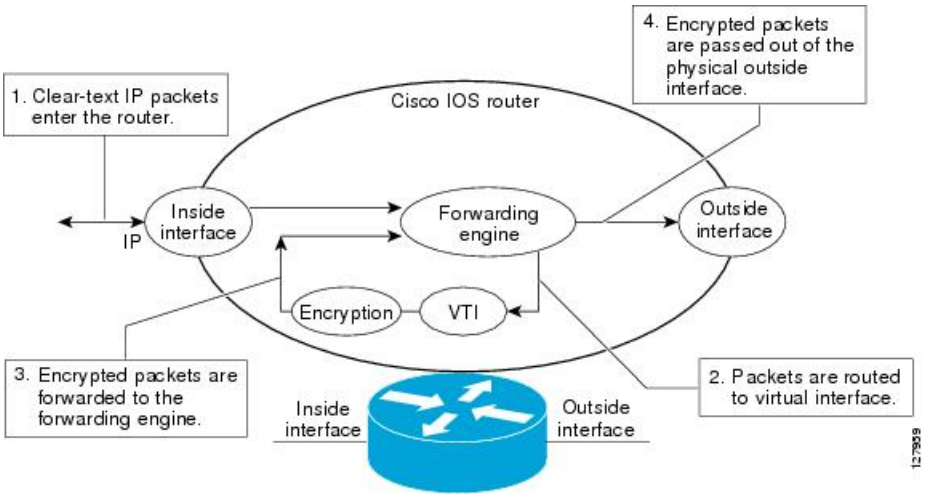
1. User 1 calls the router.
2. Router 1 authenticates User 1.
3. IPsec clones the virtual access interface from the virtual template interface.

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 . 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.

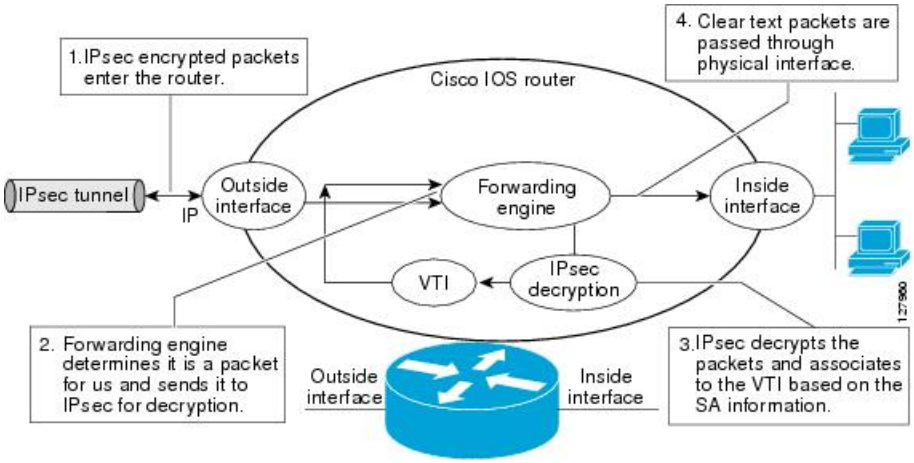
Figure 3: Packet Flow into the IPsec Tunnel



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 4: Packet Flow out of the IPsec Tunnel



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 provides benefits similar to other Cisco IOS interface.

FlexVPN Mixed Mode Support

The FlexVPN Mixed Mode feature provides support for carrying IPv4 traffic over IPsec IPv6 transport. This is the first phase towards providing dual stack support on the IPsec stack. This implementation does not support using a single IPsec security association (SA) pair for both IPv4 and IPv6 traffic.

This feature is only supported for Remote Access VPN with IKEv2 and Dynamic VTI.

The FlexVPN Mixed Mode feature provides support for carrying IPv6 traffic over IPsec IPv4 transport from Cisco IOS XE Everest 16.4.1.

Auto Tunnel Mode Support in IPsec

When configuring a VPN headend in a multiple vendor scenario, you must be aware of the technical details of the peer or responder. For example, some devices may use IPsec tunnels while others may use generic routing encapsulation (GRE) or IPsec tunnel, and sometimes, a tunnel may be IPv4 or IPv6. In the last case, you must configure an Internet Key Exchange (IKE) profile and a virtual template.

The Tunnel Mode Auto Selection feature eases the configuration and spares you about knowing the responder's details. This feature automatically applies the tunneling protocol (GRE or IPsec) and transport protocol (IPv4 or IPv6) on the virtual template as soon as the IKE profile creates the virtual access interface. This feature is useful on dual stack hubs aggregating multivendor remote access, such as Cisco AnyConnect VPN Client, Microsoft Windows7 Client, and so on.

Note The Tunnel Mode Auto Selection feature eases the configuration for a responder only. The tunnel must be statically configured for an initiator.

IPSec Mixed Mode Support for VTI

The IPSec Mixed Mode feature provides support for carrying IPv4 traffic over IPsec IPv6 transport. This is the first phase towards providing dual stack support on the IPsec stack. This implementation does not support using a single IPsec security association (SA) pair for both IPv4 and IPv6 traffic.

This feature is supported for SVTI as well as DVTI and IKEv1 as well as IKEv2.

How to Configure IPsec Virtual Tunnel Interfaces

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-number*
10. **tunnel destination** *ip-address*
11. **tunnel protection IPsec profile** *profile-name*
12. **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`	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 .
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-number* **Example:** `Device(config-if)# tunnel source loopback 0`	Specifies the tunnel source as a loopback interface.* **Note** *If you are configuring the Tunnel Mode Auto Selection feature using a virtual-template, omit the tunnel source and tunnel mode in interface virtual-template number type tunnel command. If the tunnel source and tunnel mode are specified, clients using IPv6 transport will fail to connect.
Step 10	**tunnel destination** *ip-address* **Example:**	Identifies the IP address of the tunnel destination.

	Command or Action	Purpose
	`Device(config-if)# tunnel destination 172.16.1.1`	
Step 11	**tunnel protection IPsec profile** *profile-name* **Example:** `Device(config-if)# tunnel protection IPsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 12	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring BGP over IPsec Virtual Tunnel Interfaces

Perform this task to optionally configure BGP over the virtual tunnel interfaces of two routers.

Before you begin

Perform steps in Configuring Static IPsec Virtual Tunnel Interfaces, on page 40.

SUMMARY STEPS

1. **router bgp** *autonomous-system-number*
2. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
3. **network** *network-ip-address* **mask** *subnet-mask*
4. **exit**
5. Enter the following commands on the second router.
6. **router bgp** *autonomous-system-number*
7. **neighbor** *ip-address* **remote-as** *autonomous-system-number*
8. **network** *network-ip-address* **mask** *subnet-mask*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65510`	Enters router configuration mode and creates a BGP routing process. *autonomous-system-number* —Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. Number in the range from 1 to 65535. In the example, the first router in this procedure is identified as "65510".
Step 2	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:**	*ip-address* —IP address of the adjacent router's tunnel interface.

	Command or Action	Purpose
	`Device(config-router)# neighbor 10.1.1.2 remote-as 65511`	*autonomous-system-number* —Number of an autonomous system that identifies the router of the second router. Number in the range from 1 to 65535.
Step 3	**network** *network-ip-address* **mask** *subnet-mask* **Example:** `Device(config-router)# network 2.2.2.0 mask 255.255.255.0`	*network-ip-address*—IP address of the network advertised in BGP. For example, the IP address of a loopback interface. *subnet-mask*—subnet mask of the network advertised in BGP. **Note** The BGP network command network and mask *must* exactly match a route that is already in the routing table for it to be brought into BGP and advertised to BGP neighbors. This is different from EIGRP, OSPF where the network statement just has to "cover" an interface network and it will pick up the network with mask from the interface.
Step 4	**exit** **Example:** `Device(config-router)# exit`	Exits router configuration mode.
Step 5	Enter the following commands on the second router.	
Step 6	**router bgp** *autonomous-system-number* **Example:** `Device(config)# router bgp 65511`	Enters router configuration mode and creates a BGP routing process. *autonomous-system-number* —Number of an autonomous system that identifies the router to other BGP routers and tags the routing information that is passed along. Number in the range from 1 to 65535. In the example, the second router in this procedure is identified as "65511".
Step 7	**neighbor** *ip-address* **remote-as** *autonomous-system-number* **Example:** `Device(config-router)# neighbor 10.1.1.1 remote-as 65510`	*ip-address* —IP address of the adjacent router's tunnel interface.
Step 8	**network** *network-ip-address* **mask** *subnet-mask* **Example:** `Device(config-router)# network 1.1.1.0 mask 255.255.255.0`	*network-ip-address*—IP address of the network advertised in BGP. For example, the IP address of a loopback interface. *subnet-mask*—subnet mask of the network advertised in BGP. **Note** Use the exact network IP address and subnet mask.

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* **type tunnel**
7. **tunnel mode ipsec ipv4**
8. **tunnel protection IPsec profile** *profile-name*
9. **exit**
10. **crypto isakamp profile** *profile-name*
11. **match identity address** *ip-address mask*
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.
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* **type tunnel** **Example:** `Device(config)# interface virtual-template 2 type tunnel`	Defines a virtual-template tunnel interface and enters interface configuration mode.

	Command or Action	Purpose
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* **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 ISAKMP profile to be used for the virtual template.
Step 11	**match identity address** *ip-address mask* **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 ISAKMP profile.
Step 13	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuring Multi-SA Support for Dynamic Virtual Tunnel Interfaces Using IKEv1

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. **ip vrf** *vrf-name*

4. **rd** *route-distinguisher*
5. **exit**
6. **crypto keyring** *keyring-name*
7. **pre-shared-key** *address* **key** *key*
8. **exit**
9. **crypto isakmp profile** *profile-name*
10. **keyring** *keyring-name*
11. **match identity** *address mask*
12. **virtual-template** *template-number*
13. **exit**
14. **crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*]
15. **exit**
16. **crypto ipsec profile** *name*
17. **set security-policy limit** *maximum-limit*
18. **set transform-set** *transform-set-name* [*transform-set-name2 .... transform-set-name6*]
19. **exit**
20. **interface virtual-template** *number type tunnel*
21. **ip vrf forwarding** *vrf-name*
22. **ip unnumbered** *type number*
23. **tunnel mode ipsec ipv4**
24. **tunnel protection profile ipsec** *profile-name*
25. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**ip vrf** *vrf-name* **Example:** `Device(config)# ip vrf VRF-100-1`	Defines the VRF instance and enters VRF configuration mode.
Step 4	**rd** *route-distinguisher* **Example:** `Device(config-vrf)# rd 100:21`	Creates routing and forwarding tables for a VRF.
Step 5	**exit** **Example:** `Device(config-vrf)# exit`	Exits VRF configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 6	**crypto keyring** *keyring-name* **Example:** `Device(config)# crypto keyring cisco-100-1`	Defines a crypto key ring and enters key ring configuration mode.
Step 7	**pre-shared-key** *address* **key** *key* **Example:** `Device(config-keyring)# pre-shared-key address 10.1.1.1 key cisco-100-1`	Defines the preshared key to be used for Internet Key Exchange (IKE) authentication.
Step 8	**exit** **Example:** `Device(config-keyring)# exit`	Exits keyring configuration mode and enters global configuration mode.
Step 9	**crypto isakmp profile** *profile-name* **Example:** `Device(config)# crypto isakmp profile cisco-isakmp-profile-100-1`	Defines an ISAKMP profile and enters ISAKMP configuration mode.
Step 10	**keyring** *keyring-name* **Example:** `Device(conf-isa-prof)# keyring cisco-100-1`	Configures a key ring in ISAKMP mode.
Step 11	**match identity** *address mask* **Example:** `Device(conf-isa-prof)# match identity address 10.1.1.0 255.255.255.0`	Matches an identity from the ISAKMP profile.
Step 12	**virtual-template** *template-number* **Example:** `Device(conf-isa-prof)# virtual-template 101`	Specifies the virtual template that will be used to clone virtual access interfaces.
Step 13	**exit** **Example:** `Device(conf-isa-prof)# exit`	Exits ISAKMP profile configuration mode and enters global configuration mode.
Step 14	**crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*] **Example:** `Device(config)# crypto ipsec transform-set cisco esp-aes esp-sha-hmac`	Defines the transform set and enters crypto transform configuration mode.
Step 15	**exit** **Example:** `Device(conf-crypto-trans)# exit`	Exits crypto transform configuration mode and enters global configuration mode.

	Command or Action	Purpose
Step 16	**crypto ipsec profile** *name* **Example:** `Device(config)# crypto ipsec profile cisco-ipsec-profile-101`	Defines the IPsec parameters used for IPsec encryption between two IPsec devices, and enters IPsec profile configuration mode.
Step 17	**set security-policy limit** *maximum-limit* **Example:** `Device(ipsec-profile)# set security-policy limit 3`	Defines an upper limit to the number of flows that can be created for an individual virtual access interface.
Step 18	**set transform-set** *transform-set-name* [*transform-set-name2 .... transform-set-name6*] **Example:** `Device(ipsec-profile)# set transform-set cisco`	Specifies the transform sets to be used with the crypto map entry.
Step 19	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits IPsec profile and enters global configuration mode.
Step 20	**interface virtual-template** *number type tunnel* **Example:** `Device(config)# interface virtual-template 101 type tunnel`	Creates a virtual template interface that can be configured interface and enters interface configuration mode.
Step 21	**ip vrf forwarding** *vrf-name* **Example:** `Device(config-if)# ip vrf forwarding VRF-100-1`	Associates a VRF instance with a virtual-template interface.
Step 22	**ip unnumbered** *type number* **Example:** `Device(config-if)# ip unnumbered GigabitEthernet 0.0`	Enables IP processing on an interface without assigning an explicit IP address to the interface.
Step 23	**tunnel mode ipsec ipv4** **Example:** `Device(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 24	**tunnel protection profile ipsec** *profile-name* **Example:** `Device(config-if)# tunnel protection ipsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 25	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode, and returns to privileged EXEC mode.

Configuring IPsec Mixed Mode Support for SVTIs

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. Do one of the following:
   - **tunnel mode ipsec ipv4 v6-overlay**
   - **tunnel mode ipsec ipv6 v4-overlay**
9. **tunnel source** *interface-type interface-type*
10. **tunnel destination** *ip-address*
11. **tunnel protection IPsec profile** *profile-name*
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.
Step 5	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits IPsec profile configuration mode, and enters global configuration mode.

	Command or Action	Purpose
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	Do one of the following: • **tunnel mode ipsec ipv4 v6-overlay** • **tunnel mode ipsec ipv6 v4-overlay** **Example:** `Device(config-if)# tunnel mode ipsec ipv4 v6-overlay`	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* **Example:** `Device(config-if)# tunnel protection IPsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 12	**end** **Example:** `Device(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring IPsec Mixed Mode Support for Dynamic VTIs

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec profile** *profile-name*

4. **set mixed mode**
5. **set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*]
6. **exit**
7. **interface virtual-template** *number* **type tunnel**
8. **tunnel mode ipsec ipv4**
9. **tunnel protection IPsec profile** *profile-name*
10. **exit**
11. **crypto isakamp profile** *profile-name*
12. **match identity address** *ip-address mask*
13. **virtual template** *template-number*
14. **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 mixed mode** **Example:** `Device(config)# set mixed mode`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec devices and enters IPsec profile configuration mode.
Step 5	**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 6	**exit** **Example:** `Device(ipsec-profile)# exit`	Exits ipsec profile configuration mode and enters global configuration mode.
Step 7	**interface virtual-template** *number* **type tunnel** **Example:** `Device(config)# interface virtual-template 2 type tunnel`	Defines a virtual-template tunnel interface and enters interface configuration mode.

	Command or Action	Purpose
Step 8	**tunnel mode ipsec ipv4** **Example:** `Device(config-if)# tunnel mode ipsec ipv4`	Defines the mode for the tunnel.
Step 9	**tunnel protection IPsec profile** *profile-name* **Example:** `Device(config-if)# tunnel protection ipsec profile PROF`	Associates a tunnel interface with an IPsec profile.
Step 10	**exit** **Example:** `Device(config-if)# exit`	Exits interface configuration mode.
Step 11	**crypto isakamp profile** *profile-name* **Example:** `Device(config)# crypto isakamp profile profile1`	Defines the ISAKMP profile to be used for the virtual template.
Step 12	**match identity address** *ip-address mask* **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 13	**virtual template** *template-number* **Example:** `Device(config)# virtual-template 1`	Specifies the virtual template attached to the ISAKMP profile.
Step 14	**end** **Example:** `Device(config)# end`	Exits global configuration mode and enters privileged EXEC mode.

Configuring Multi-SA Support for Static IPsec Virtual Tunnel Interfaces

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 the transform sets that can be used.

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** | **ipv6**}

Example:

```
Device(config-if)# tunnel mode ipsec ipv4
```

Defines the mode for the tunnel.

Step 9 **tunnel source** *interface-type interface-number*

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 policy** {**ipv4** | **ipv6**} *acl*

Example:

```
Device(config-if)# tunnel protection ipsec policy ipv4 ipsec-acl1
```

Associates an ACL with an SVTI to define non-any-any traffic selectors.

Step 12 **tunnel protection ipsec profile** *profile-name*

Example:

```
Device(config-if)# tunnel protection IPsec profile PROF
```

Associates a tunnel interface with an IPsec profile.

Step 13 **exit**

Example:

```
Device(config-if)# exit
```

Exits interface configuration mode and enters global configuration mode.

Step 14 **ip access-list extended** *name* OR **ipv6 access-list** *name*

Example:

IPv4:

```
Device(config)# ip access-list extended ipsec-acl1
```

IPv6:

```
Device(config)# ipv6 access-list ipsec-acl1
```

Defines an extended IP access list using a name and enters extended named access list configuration mode.

Step 15 **permit** *protocol source* [*source-wildcard*] *destination* [*destination-wildcard*] [**option** *option-name*]

Example:

```
Device(config-ext-nacl)# permit ip 30.0.1.0 0.0.0.255 10.0.1.0 0.0.0.255
```

Permits traffic that matches all of the conditions specified in the statement.

Do not use the keyword **any** as the wildcard for both the source and destination proxies. For the ‘any any’ traffic selector, use the default SVTI without an attached ACL.

Do not use **deny** statements.

Step 16 **end**

Example:

```
Device(config-ext-nacl)# end
```

Exits standard named access list configuration mode and enters privileged EXEC mode.

Configuring Tunnel Mode as Dual-overlay

To configure the tunnel mode as dual-overlay, perform these steps:

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 tunnel** *type number*

Example:

```
Device(config)# interface tunnel 1
```

Specifies a tunnel interface and number, and enters interface configuration mode.

Step 4 **ipv6 enable**

Example:

```
Device(config-if)# ipv6 enable
```

Enables IPv6 processing on an interface that has not been configured with an explicit IPv6 address.

Step 5 **tunnel source** { **ipv4-address | interface-type | interface-number**}

Example:

```
Device(config-if)# tunnel source Gigabitethernet 1
```

Specifies the source IPv6 address or the source interface type and number for the tunnel interface. If an interface type and number are specified, that interface must be configured with an IPv6 address.

Step 6 **tunnel mode ipsec dual-overlay**

Example:

```
Device(config-if)# tunnel mode ipsec dual-overlay
```

Specifies a dual-overlay tunnel. The **tunnel mode ipsec dual-overlay** command specifies the encapsulation protocol for the tunnel.

Step 7 **tunnel destination ip address** *address mask*

Example:

```
Device(config-if)# tunnel destination 89.89.89.1 255.255.255.255.0
```

Specifies the destination IPv6 address for the tunnel interface.

Step 8 **tunnel protection ipsec profile** *ipserc profile-name*

Example:

```
Device(config-if)# tunnel protection IPsec profile ipsecprof
```

Associates a tunnel interface with an IPsec profile. The *name* argument specifies the name of the IPsec profile; this value must match the *name* specified in the **crypto IPsec profile** *name* command

Step 9 **exit**

Example:

```
Device(config-if)# exit
```

Exits interface configuration mode and enters global configuration mode.

Step 10 **end**

Example:

```
Device(config-if)# end
```

Exits interface configuration mode and returns to 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 5: VTI with IPsec

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

 load-interval 30
 tunnel source 10.0.149.203
 tunnel destination 10.0.149.217
 tunnel mode IPsec ipv4
 tunnel protection IPsec profile P1
!

 ip address 10.0.149.203 255.255.255.0
 duplex full
!

 ip address 10.0.35.203 255.255.255.0
```

```
 duplex full
!
ip classless
ip route 10.0.36.0 255.255.255.0 Tunnel0
line con 0
line aux 0
line vty 0 4
end
```

Router 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

 tunnel source 10.0.149.217
 tunnel destination 10.0.149.203
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile P1
!
interface
 ip address 10.0.149.217 255.255.255.0
 speed 100
 full-duplex
!
interface
 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

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 IPsec Static Virtual Tunnel Interface

```
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,
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.

C8000 Router Configuration

```
hostname c8000
```

```
.
.
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.

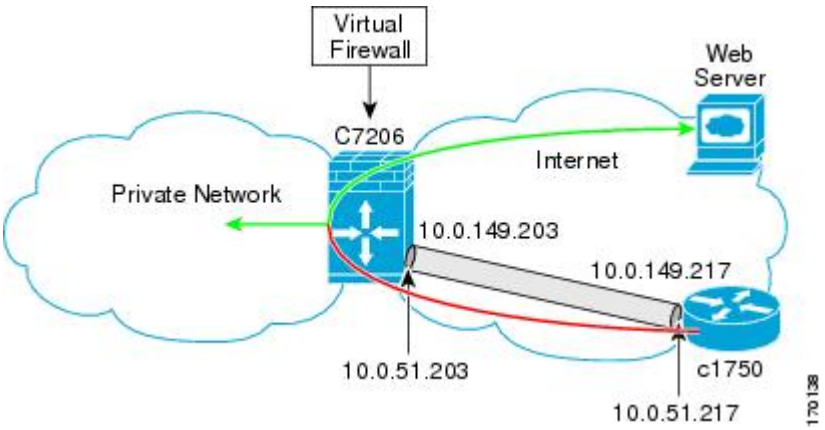
C8000 Router Configuration

```
hostname c8000
.
.
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 6: Static VTI with Virtual Firewall



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

C8000 Router Configuration

```
hostname c8000
.
.
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.

C8000 Router Configuration

```
hostname c8000
!
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

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

The following example shows how to configure VRF-aware IPsec under a virtual template to take advantage of the DVTI:

```
hostname c8000
!
ip vrf VRF-100-1
```

```
  rd 1:1
!
ip vrf VRF-100-2
  rd 1:1
!
!
!
crypto keyring cisco-100-1
  pre-shared-key address 10.1.1.1 key cisco-100-1
crypto keyring cisco-100-2
  pre-shared-key address 10.1.2.1 key cisco-100-2
crypto isakmp profile cisco-isakmp-profile-100-1
  keyring cisco-100-1
  match identity address 10.1.1.0 255.255.255.0
  virtual-template 101
crypto isakmp profile cisco-isakmp-profile-100-2
  keyring cisco-100-2
  match identity address 10.1.2.0 255.255.255.0
  virtual-template 102
!
!
crypto ipsec transform-set cisco esp-aes esp-sha-hmac
!
crypto ipsec profile cisco-ipsec-profile-101
  set security-policy limit 3
  set transform-set cisco
!
crypto ipsec profile cisco-ipsec-profile-102
  set security-policy limit 5
  set transform-set Cisco
!
interface Virtual-Template101 type tunnel
  ip vrf forwarding VRF-100-1
  ip unnumbered Ethernet 0/0
  tunnel mode ipsec ipv4
  tunnel protection ipsec profile cisco-ipsec-profile-101
!
interface Virtual-Template102 type tunnel
  ip vrf forwarding VRF-100-2
  ip unnumbered Ethernet 0/0
  tunnel mode ipsec ipv4
  tunnel protection ipsec profile cisco-ipsec-profile-102
!
```

Example: VRF-Aware IPsec with Dynamic VTI When VRF Is Configured Under a Virtual Template with the Gateway Option in an IPsec Profile

The following example shows how to configure VRF-aware IPsec to take advantage of the DVTI, when the VRF is configured under a virtual template with the gateway option in an IPsec profile.

```
hostname c8000
!
ip vrf VRF-100-1
 rd 1:1
!
ip vrf VRF-100-2
 rd 1:1
!
!
!
```

```
crypto keyring cisco-100-1
 pre-shared-key address 10.1.1.1 key cisco-100-1
crypto keyring cisco-100-2
 pre-shared-key address 10.1.2.1 key cisco-100-2
crypto isakmp profile cisco-isakmp-profile-100-1
 keyring cisco-100-1
 match identity address 10.1.1.0 255.255.255.0
 virtual-template 101
crypto isakmp profile cisco-isakmp-profile-100-2
 keyring cisco-100-2
 match identity address 10.1.2.0 255.255.255.0
 virtual-template 102
!
!
crypto ipsec transform-set cisco esp-3des esp-sha-hmac
!
crypto ipsec profile cisco-ipsec-profile-101
 set security-policy limit 3
 set transform-set cisco
 set reverse-route gateway 172.16.0.1
!
crypto ipsec profile cisco-ipsec-profile-102
 set security-policy limit 5
 set transform-set cisco
 set reverse-route gateway 172.16.0.1
!
interface Virtual-Template101 type tunnel
 ip vrf forwarding VRF-100-1
 ip unnumbered Ethernet 0/0
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile cisco-ipsec-profile-101
!
interface Virtual-Template102 type tunnel
 ip vrf forwarding VRF-100-2
 ip unnumbered Ethernet 0/0
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile cisco-ipsec-profile-102
!
```

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

```
hostname c8000
!
ip vrf VRF-100-1
  rd 1:1
!
ip vrf VRF-100-2
  rd 1:1
!
crypto keyring cisco-100-1
  pre-shared-key address 10.1.1.1 key cisco-100-1
crypto keyring cisco-100-2
  pre-shared-key address 10.1.2.1 key cisco-100-2
crypto isakmp profile cisco-isakmp-profile-100-1
  vrf VRF-100-1
  keyring cisco-100-1
  match identity address 10.1.1.0 255.255.255.0
  virtual-template 1
```

```
crypto isakmp profile cisco-isakmp-profile-100-2
  vrf VRF-100-2
  keyring cisco-100-2
  match identity address 10.1.2.0 255.255.255.0
  virtual-template 1
!
!
crypto ipsec transform-set cisco esp-aes esp-sha-hmac
crypto ipsec profile cisco-ipsec-profile
  set security-policy limit 3
  set transform-set cisco
!
!
!
interface Virtual-Template 1 type tunnel
  ip unnumbered ethernet 0/0
  tunnel mode ipsec ipv4
  tunnel protection ipsec profile cisco-ipsec-profile
!
!
```

Example: VRF-Aware IPsec with a Dynamic VTI When VRF Is Configured Under an ISAKMP Profile and a Gateway Option in an IPsec Profile

The following example shows how to configure VRF-aware IPsec to take advantage of the DVTI, when the VRF is configured under an ISAKMP profile and a gateway option in an IPsec profile:

```
hostname C8000 server
!
ip vrf VRF-100-1
 rd 1:1
!
ip vrf VRF-100-2
 rd 1:1
!
crypto keyring cisco-100-1
 pre-shared-key address 10.1.1.1 key cisco-100-1
crypto keyring cisco-100-2
 pre-shared-key address 10.1.2.1 key cisco-100-2
crypto isakmp profile cisco-isakmp-profile-100-1
 vrf VRF-100-1
 keyring cisco-100-1
 match identity address 10.1.1.0 255.255.255.0
 virtual-template 1
crypto isakmp profile cisco-isakmp-profile-100-2
 vrf VRF-100-2
 keyring cisco-100-2
 match identity address 10.1.2.0 255.255.255.0
 virtual-template 1
!
!
crypto ipsec transform-set cisco esp-3des esp-sha-hmac
crypto ipsec profile cisco-ipsec-profile
 set security-policy limit 3
 set transform-set cisco
 set reverse-route gateway 172.16.0.1
!
!
```

```
!
interface Virtual-Template1 type tunnel
 ip unnumbered Ethernet 0/0
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile cisco-ipsec-profile
!
!
```

Example: VRF-Aware IPsec with a Dynamic VTI When a VRF Is Configured Under Both a Virtual Template and an ISAKMP Profile

Note When separate VRFs are configured under an ISAKMP profile and a virtual template, the VRF configured under the virtual template takes precedence. This configuration is not recommended.

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

```
hostname C8000 server
.
.
.
ip vrf test-vti2
 rd 1:2
 route-target export 1:1
 route-target import 1:1
!
.
.
.
ip vrf test-vti1
 rd 1:1
 route-target export 1:1
 route-target import 1:1
!
.
.
.
crypto isakmp profile cisco-isakmp-profile
 vrf test-vti2
 keyring key
 match identity address 10.1.1.0 255.255.255.0
!
.
.
.
interface Virtual-Template1 type tunnel
 ip vrf forwarding test-vti1
 ip unnumbered Loopback 0
 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 c8000
.
.
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.

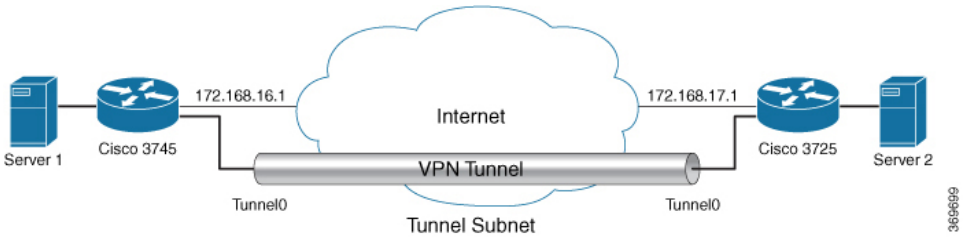
```
hostname c8000
.
.
class-map match-all VTI
 match any
!
policy-map VTI
  class VTI
  police cir 2000000
    conform-action transmit
    exceed-action drop
!
.
.
interface Virtual-Template1 type tunnel
 ip vrf forwarding test-vti1
 ip unnumbered Loopback0
 ip virtual-reassembly
 tunnel mode ipsec ipv4
 tunnel protection ipsec profile test-vti1
 service-policy output VTI
!
.
.
!
end
```

Example: Static Virtual Tunnel Interface with Multiple IPsec SAs

In the following examples an IPSec tunnel is to be established between two routers Cisco 3745 and Cisco 3725 using SVTI. The configuration uses non-any-any traffic selectors and enables the formation of multiple IPSec SAs.

Sample configuration on a Router with the IPv4 Tunnel Mode:

The following figure illustrates the reference topology for the configuration.



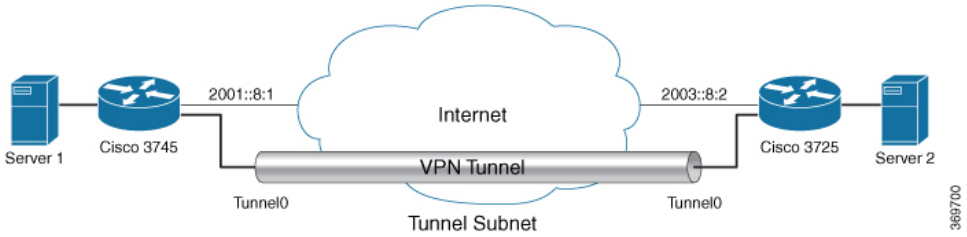
Sample configuration for the router Cisco 3745 is as follows:

```
crypto isakmp policy 1
 authentication pre-share
 group 2
!
crypto isakmp policy 5
```

```
 encr 3des
 authentication pre-share
 group 2
crypto isakmp key example address 172.168.17.1
!
!
crypto ipsec transform-set svti1 esp-3des esp-sha-hmac
 mode tunnel
!
!
crypto ipsec profile ipsec_prof
 set transform-set svti1
!
!
!
interface Loopback0
 ip address 30.0.0.1 255.255.255.0
!
interface Loopback1
 ip address 50.0.0.1 255.255.255.0
!
interface Tunnel0
 ip address 11.1.1.2 255.255.255.0
 tunnel source Ethernet0/0
 tunnel mode ipsec ipv4
 tunnel destination 172.168.17.1
 tunnel protection ipsec policy ipv4 ipsec_acl1
 tunnel protection ipsec profile ipsec_prof
!
interface Ethernet0/0
 ip address 172.168.16.1 255.255.255.0
!
!
ip access-list extended ipsec_acl1
permit ip 30.0.0.0 0.0.0.255 40.0.0.0 0.0.0.255
permit ip 50.0.0.0 0.0.0.255 60.0.0.0 0.0.0.255
```

Sample configuration on a Router with the IPv6 Tunnel Mode:

The following figure illustrates the reference topology for the configuration.



Sample configuration for the router Cisco 3745 is as follows:

```
crypto isakmp policy 1
 authentication pre-share
 group 2
!
crypto isakmp policy 5
 encr 3des
 authentication pre-share
 group 2
crypto isakmp key example address ipv6 2003::8:2/112
!
!
```

```
crypto ipsec transform-set svti1 esp-3des esp-sha-hmac
 mode tunnel
!
!
crypto ipsec profile ipsec_prof
 set transform-set svti1
!
!
!
interface Loopback0
 ipv6 address 2005::10:1/112
 ipv6 enable
!
interface Loopback1
 ipv6 address 2005::15:1/112
 ipv6 enable
!
interface Loopback2
 ipv6 address 2005::20:1/112
 ipv6 enable
!
interface Tunnel0
 ip address 11.1.1.2 255.255.255.0
 ipv6 address 400::10:1/112
 ipv6 enable
 tunnel source Ethernet0/0
 tunnel mode ipsec ipv6
 tunnel destination 2003::8:2
 tunnel protection ipsec policy ipv6 ipsec_acl2
 tunnel protection ipsec profile ipsec_prof
!
interface Ethernet0/0
 ipv6 address 2001::8:1/112
 ipv6 enable
!
!
ipv6 access-list ipsec_acl2
sequence 10 permit ipv6 host 2005::10:1 host 2005::11:1
sequence 20 permit ipv6 host 2005::15:1 host 2005::16:1
sequence 30 permit ipv6 host 2005::20:1 host 2005::21:1
```

Example: Configuring Tunnel Mode as Dual-overlay

The following example shows how to configure tunnel mode as dual-overlay:

```
Device# configure terminal
Router(config)# interface tunnel 1
Router(config-if)# ipv6 enable
Router(config-if)# tunnel source ethernet 0/0
Router(config-if)# tunnel mode ipsec dual-overlay
Router(config-if)# tunnel destination 89.89.89.1 255.255.255.255.0
Device(config-if)# tunnel protection IPsec profile ipsecprof
```

Verifying the Tunnel Mode as Dual-overlay Configuration

Use the following commands to troubleshoot your configuration:

- **Show crypto session [detail]**
- **Show crypto ipsec sa**
- **Show crypto map**

- **Show crypto socket**
- **Show crypto ikev2 session [detail]**

```
Device# show crypto map
Crypto Map: "Tunnel0-head-0" IKEv2 profile: prof

Crypto Map IPv4 "Tunnel0-head-0" 65536 ipsec-isakmp
 IKEv2 Profile: prof
 Profile name: prof
 Security association lifetime: 4608000 kilobytes/120 seconds
 Dualstack (Y/N): N

 Responder-Only (Y/N): N
 PFS (Y/N): N
 Mixed-mode : Disabled
 Transform sets={
  default:  { esp-aes esp-sha-hmac  } ,
 }

Crypto Map IPv4 "Tunnel0-head-0" 65537 ipsec-isakmp
 Map is a PROFILE INSTANCE.
 Peer = 10.10.10.2
 IKEv2 Profile: prof
 Extended IP access list
     access-list  permit ip any any
 Current peer: 10.10.10.2
 Security association lifetime: 4608000 kilobytes/120 seconds
        Dualstack (Y/N): Y
        TRUE  ident (addr/mask/prot/port): {LOCAL -> REMOTE}
             0.0.0.0/0.0.0.0/0/0 -> 0.0.0.0/0.0.0.0/0/0
             ::/0.0.0.0/0/0 -> ::/0/0/0
 Responder-Only (Y/N): N
 PFS (Y/N): N
 Mixed-mode : Disabled
 Transform sets={
  default:  { esp-aes esp-sha-hmac  } ,
 }
 Always create SAs
 Interfaces using crypto map Tunnel0-head-0:
  Tunnel0

Device# show  crypto ipsec sa

interface: Tunnel0
    Crypto map tag: Tunnel0-head-0, local addr 10.10.10.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
   remote ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
    TRUE  ident (addr/mask/prot/port): {LOCAL -> REMOTE}
             0.0.0.0/0.0.0.0/0/0 -> 0.0.0.0/0.0.0.0/0/0
             ::/0.0.0.0/0/0 -> ::/0/0/0
   current_peer 10.10.10.2 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 10.10.10.1, remote crypto endpt.: 10.10.10.2
     plaintext mtu 1438, path mtu 1500, ip mtu 1500, ip mtu idb Ethernet0/0
```

```
     current outbound spi: 0x4776A36B(1198957419)
     PFS (Y/N): N, DH group: none

     inbound esp sas:
      spi: 0xA97EDEE7(2843664103)
        transform: esp-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 4, flow_id: 4, sibling_flags FFFFFFFF80000040, crypto map: Tunnel0-head-0
         sa timing: remaining key lifetime (k/sec): (4377587/76)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0x4776A36B(1198957419)
        transform: esp-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3, flow_id: 3, sibling_flags FFFFFFFF80000040, crypto map: Tunnel0-head-0
         sa timing: remaining key lifetime (k/sec): (4377587/76)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE(ACTIVE)

     outbound ah sas:

     outbound pcp sas:
```

```
Device# show crypto socket

Number of Crypto Socket connections 1

   Tu0 Peers (local/remote): 10.10.10.1/10.10.10.2
       Local Ident  (addr/mask/port/prot): (0.0.0.0/0.0.0.0/0/0)
       Remote Ident (addr/mask/port/prot): (0.0.0.0/0.0.0.0/0/0)
        TRUE  ident (addr/mask/prot/port): {LOCAL -> REMOTE}
             0.0.0.0/0.0.0.0/0/0 -> 0.0.0.0/0.0.0.0/0/0
             ::/0.0.0.0/0/0 -> ::/0/0/0
       IPSec Profile: "prof"
       Socket State: Open
       Client: "TUNNEL SEC" (Client State: Active)
Crypto Sockets in Listen state:
Client: "TUNNEL SEC" Profile: "prof" Map-name: "Tunnel0-head-0"
```

```
Device# show cry ikev2 session
 IPv4 Crypto IKEv2 Session

Session-id:1, Status:UP-ACTIVE, IKE count:1, CHILD count:1

Tunnel-id Local                 Remote                fvrf/ivrf            Status
1         10.10.10.1/500        10.10.10.2/500        none/none            READY
      Encr: AES-CBC, keysize: 256, PRF: SHA512, Hash: SHA512, DH Grp:19, Auth sign: PSK,
Auth verify: PSK
      Life/Active Time: 86400/145 sec
      CE id: 1001, Session-id: 1
      Local spi: 25A0B173944015D3       Remote spi: 9F0C7677425670E1
Child sa:
          local selector  0.0.0.0/0 - 255.255.255.255/65535
          local selector  ::/0 - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF/65535
          remote selector 0.0.0.0/0 - 255.255.255.255/65535
          remote selector ::/0 - FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF/65535
          ESP spi in/out: 0xA97EDEE7/0x4776A36B
```

```
 IPv6 Crypto IKEv2 Session

Device# show crypto session
Crypto session current status

Interface: Tunnel0
Profile: prof
Session status: UP-ACTIVE
Peer: 10.10.10.2 port 500
  Session ID: 1
  IKEv2 SA: local 10.10.10.1/500 remote 10.10.10.2/500 Active
  IPSEC FLOW: permit ip   0.0.0.0/0.0.0.0 0.0.0.0/0.0.0.0
  TRUE IDENT (addr/mask/prot/port): {LOCAL -> REMOTE}
             0.0.0.0/0.0.0.0/0/0 -> 0.0.0.0/0.0.0.0/0/0
             ::/0.0.0.0/0/0 -> ::/0/0/0
        Active SAs: 2, origin: crypto map
```

Additional References for IPsec Virtual Tunnel Interface

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command 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*
EasyVPN 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*

Standard/RFC	Title
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.

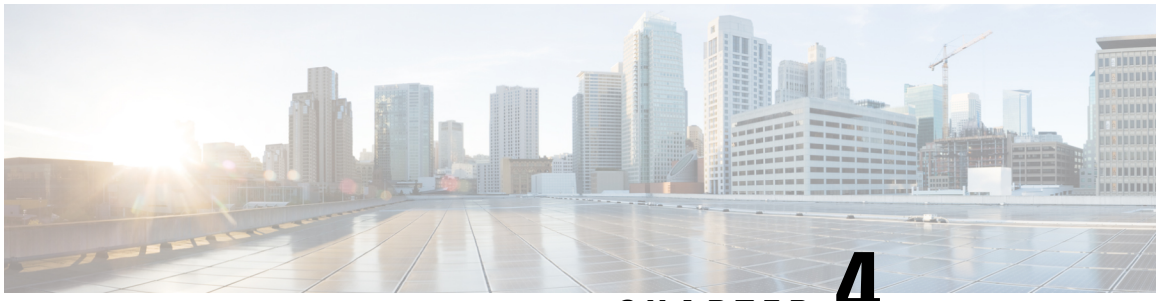
Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 3: Feature Information for IPsec Virtual Tunnel Interfaces

Feature Name	Releases	Feature Configuration Information
Dynamic IPsec VTIs		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
FlexVPN Mixed Mode Support		The FlexVPN Mixed Mode feature provides support for carrying IPv4 traffic over IPsec IPv6 transport. This is the first phase towards providing dual stack support on the IPsec stack. This implementation does not support using a single IPsec security association (SA) pair for both IPv4 and IPv6 traffic. This feature is only supported for Remote Access VPN with IKEv2 and Dynamic VTI.
Multi-SA for Dynamic VTIs		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		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.
Tunnel Mode Auto Selection		The Tunnel Mode Auto Selection feature eases the configuration and spares you about knowing the responder's details. This feature automatically applies the tunneling protocol (GRE or IPsec) and transport protocol (IPv4 or IPv6) on the virtual template as soon as the IKE profile creates the virtual access interface. The following command was introduced or modified: **virtual-template**

Feature Name	Releases	Feature Configuration Information
FlexVPN Mixed Mode v6 over v4 Transport		The FlexVPN Mixed Mode v6 over v4 Transport feature provides support for carrying IPv6 traffic over IPsec IPv4 transport. This implementation does not support using a single IPsec security association (SA) pair for both IPv4 and IPv6 traffic.



CHAPTER 4

Session Initiation Protocol Triggered VPN

Session Initiation Protocol Triggered VPN (SIP-Triggered VPN or VPN-SIP) is a service offered by service providers where a VPN is set up using Session Initiation Protocol (SIP) for on-demand media or application sharing between peers. The VPN-SIP feature defines the process in which two SIP user agents resolve each other's IP addresses, exchange the fingerprints of their self-signed certificates, third-party certificates, or pre-shared key securely, and agree to establish an IPsec-based VPN.

Service providers offer the VPN-SIP service to their customers that have SIP-based services such as bank ATMs or branches. This VPN-SIP service replaces an ISDN connection for backup network functionality. If the primary broadband service link goes down, these bank ATMs or branches connect to their central headend or data centres through the VPN-SIP service.

The SIP server of the service provider, which coordinates the VPN-SIP service, is also used for billing of the service based on the time the service is used.

Information about VPN-SIP

Components for VPN-SIP Solution

VPN-SIP uses IPSec Static Virtual Tunnel Interface (SVTI). IPSec SVTI stays in active (UP) state even when there is no IPSec security association (SA) established between the tunnel interface and the SVTI peer.

The following are three components for the VPN-SIP Solution:

- SIP
- VPN-SIP

- Crypto (IP Security (IPsec), Internet Key Exchange (IKE), Tunnel Protection (TP), Public Key Infrastructure (PKI) modules within crypto)

Sesssion Initiation Protocol

SIP is used as a name resolution mechanism to initiate an IKE session. VPN-SIP uses SIP service to establish a VPN connection to a home or a small business router that does not have a fixed IP address. This connection is achieved using self-signed certificates or pre-shared keys. SIP negotiates the use of IKE for media sessions in the Session Description Protocol (SDP) offer-and-answer model.

SIP is statically configured. One tunnel interface must be configured for each remote SIP number.

SIP also provides billing capabilities for service providers to charge customers based on the SIP number, for using the VPN-SIP service. Billing based on SIP numbers happens in the service provider network and is independent of the end devices like Cisco VPN-SIP routers.

VPN-SIP Solution

VPN-SIP is the central block that coordinates between SIP and Crypto modules, and provides an abstraction between them.

When traffic destined to a remote network behind a SIP number is routed to the tunnel interface, the IPSec control plane gets a trigger from packet switching path as there is no IPSEC SA configured to that peer. IPsec control plane passes the trigger to VPN-SIP as the tunnel is configured for VPN-SIP.

Note Static routes for remote networks for that SIP number must be configured to point to that tunnel interface.

When the VPN-SIP service is triggered, SIP sets up the call with a SIP phone number pair. SIP also passes incoming call details to the VPN-SIP and negotiates IKE media sessions using local address and fingerprint information of the local self-signed certificate or pre-shared key. SIP also passes remote address and fingerprint information to VPN-SIP.

The VPN-SIP service listens to tunnel status updates and invokes SIP to tear down the SIP session. The VPN-SIP service also provides a means to display current and active sessions.

Feature at a glance

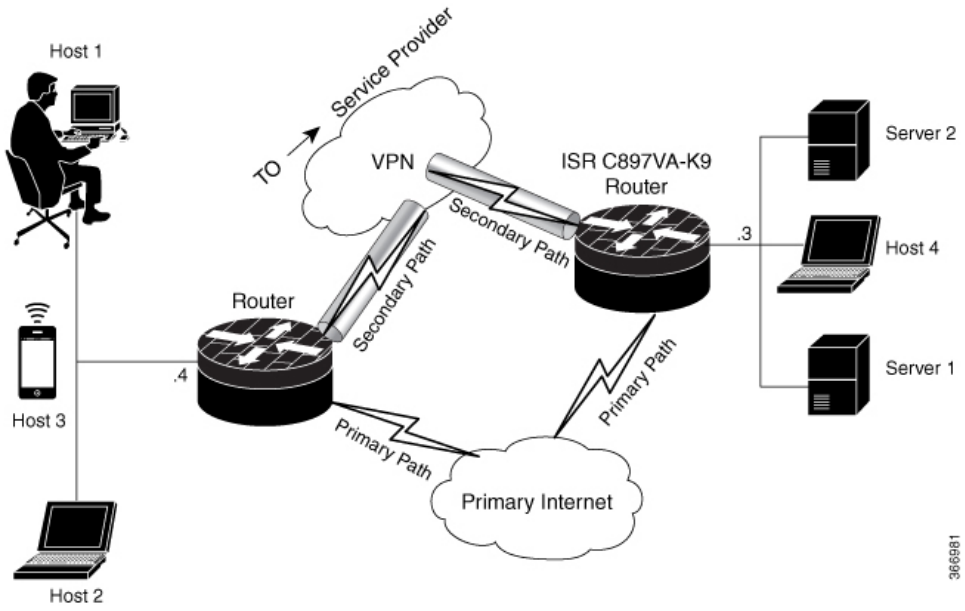
The following steps summarize how the VPN-SIP feature works:

- IP SLA monitors the primary link using route tracking. When the primary link fails IP SLA detects this failure.
- Once the primary path fails, IP SLA switches the default route to the higher metric route that is configured on the router.
- When relevant traffic tries to flow using the secondary link, SIP sends an invite message to the SIP server to obtain the VPN peer information.
- The router receives the VPN peer information (IP address, local and remote SIP numbers, IKE port, and finger print) and it establishes VPN-SIP tunnel.

- When the primary path comes back up, IP SLA detects the primary path and the route falls back to the original path. When the idle timer expires, IPSec is torn down and a SIP call is disconnected.

Following is the topology for the VPN-SIP solution:

Figure 7: VPN-SIP Topology



SIP Call Flow

The SIP call flow is divided into initiation at the local peer and call receipt at the remote peer.

At SIP Call Intitiation

When packets are routed to an SVTI interface in data plane, the SIP call must be placed to the peer SIP number to resolve its address, so that VPN tunnel can be brought up.

- When local auth-type is PSK, IKEv2 finds the matching key for a peer SIP number. The IKEv2 keyring must be configured with id_key_id type (string) as SIP number for each SIP peer. IKEv2 computes the fingerprint of the looked-up key and passes it to VPN-SIP.
- When local auth-type is a self-signed certificate or an third-party certificate, IKEv2 computes the fingerprint of the local certificate configured under the IKEv2 profile and passes it to the VPN-SIP

The VPN-SIP module interacts with SIP to setup SIP call to the peer. When the call is successful, VPN-SIP sets the tunnel destination of SVTI to the resolved IP address, requesting SVTI to initiate the VPN tunnel.

Note When a wildcard key is required, use the authentication local pre-share key command and the authentication remote pre-share key command in IKEv2 profile.

When SIP call is received at the remote peer

When a SIP call is received from a peer, following interactions occur between various crypto modules:

- The Tunnel Protection helps VPN-SIP module to set tunnel destination address.
- IKEv2 returns local auth-type (PSK or PKI) and local fingerprint to the VPN-SIP module. When local auth-type is PSK, IKEv2 finds a matching key for a corresponding SIP number.

Note IKEv2 only knows peer by its SIP number.

During the SIP call negotiation between peers, each peer must select a unique local IKEv2 port number to be exchanged over the SDP. To support different port numbers for each session, the VPN-SIP module programmatically configures IP Port Address Translation (PAT) to translate between IKEv2 port (4500) and the port number exchanged over SDP. For the translation to work IP NAT must be configured on secondary link and the loopback interface configured as the VPN-SIP tunnel source. The lifetime of the translation is limited to the lifetime of the VPN-SIP session.

SDP Offer and Answer

Following is the sample for SDP offer and answer that is negotiated in the SIP call as defined in RFC 6193:

```
offer SDP
      ...
      m=application 50001 udp ike-esp-udpencap
      c=IN IP4 10.6.6.49
      a=ike-setup:active
      a=fingerprint:SHA-1 \
      b=AS:512
      4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
      ...

   answer SDP
      ...
      m=application 50002 udp ike-esp-udpencap
      c=IN IP4 10.6.6.50
      a=ike-setup:passive
      a=fingerprint:SHA-1 \
      b=AS:512
      D2:9F:6F:1E:CD:D3:09:E8:70:65:1A:51:7C:9D:30:4F:21:E4:4A:8E
```

As part of the SDP negotiation, both peers negotiate the maximum bandwidth rate for the VPN-SIP session using the b=AS :number SDP attribute. if the peers mention different bandwidth numbers in their SDP, both of them should honor the minimum value as the maximum bandwidth. If b=AS :number SDP attribute is missing in the offer or answer, the SIP call is not successfully set up.

The negotiated maximum bandwidth is applied on the SVTI tunnel interface through the programmatically configured QoS policy in the output direction. The programmatically configured QoS policy is not applied and session fails, if there is a pre-existing statically configured policy.

Once SIP call is complete and address of the peer is resolved, VPN-SIP sets tunnel destination of SVTI and sends a request to initiate tunnel.

IKEv2 Negotiation

Following is the process for IKEv2 Security Session (SA) negotiation:

- Before starting the session, IKEv2 checks with VPN-SIP if the session is a VPN-SIP session.
- If it's a VPN-SIP session and local auth-type is PSK, IKEv2 looks up the PSK key pair using SIP number of the peer instead of IP address of the peer.
- For validating self-signed certificate, IKEv2 checks if the certificate is self-signed and validates the certificate.
    - In addition to existing AUTH payload validation as part of IKEv2 protocol, IKEv2 calculates hash of the received certificate or looked-up PSK and compares with the fingerprint from SIP negotiation that IKEv2 queries from VPN-SIP module. Only if the fingerprint matches, IKEv2 considers authentication of peer is valid. If not, IKEv2 declares that peer has failed to authenticate and fails the VPN session.

VPN-SIP solution depends on IPSEC idle timer to detect that traffic is no longer routed over the backup VPN. The idle-time configuration under the IPSec Profile is mandatory for session to be disconnected when there is no traffic. 120 seconds is the recommended time.

VPN-SIP and SIP coordinate to tear down SIP call.

When IPsec idle time expires the VPN-SIP module informs the IKEv2 to bring down the IPsec tunnel. VPN-SIP requests the SIP module to disconnect the SIP call, without waiting for confirmation from the IKEv2.

When SIP call disconnect is received from the peer, VPN-SIP module informs the IKEv2 to bring down the IPsec tunnel, and acknowledges to SIP to tear down the SIP call.

Supported Platforms

The VPN-SIP feature is supported on the following platforms:

Prerequisites for VPN-SIP

- Security K9 license must be enabled on the router.
- The routers must have a minimum memory of 1 GB.
- For the SIP register request of the SIP User Agent to succeed, the SIP registrar must be available to the VPN-SIP routers.
- The DHCP server must support option 120 and 125 to obtain the SIP server address, which is needed for registration and establishing the SIP session.
- Proper routing configurations must be completed to ensure backup WAN path is used when primary path is down.
- Maximum Transmission Unit (MTU) of the tunnel interface must be less than the MTU of the secondary WAN interface.
- When self-signed or third-party certificates are used for IKEv2 authentication, configure IKEv2 fragmentation on the VPN-SIP router to avoid fragmentation at the IP layer.
- NAT SIP ALG must be disabled.
- Caller ID notification service must be configured in the network.

Restrictions for VPN-SIP

- VPN-SIP and CUBE/SIP gateway cannot be configured on the same device. When CUBE license is active on the device, only CUBE will be functional.
- Only IPv4 is supported for transport and media (IPv4 transport for SIP registration, SIP signaling, and IPv4 packets encrypted over IPv4 transport).
- SIP signalling with peer devices behind NAT is not supported (ICE and STUN are not supported.
- SIP negotiation is supported only in global VRF.
- Remote-access VPN features like private address assignment, configuration mode exchange (CP payloads), routes exchange, are not supported.
- Routing protocols over the VPN-SIP session are not supported.
- Only Rivest-Shamir-Addleman (RSA) server self-signed certificates are supported.
- Pre-shared key lookup functionality using authentication, authorization, and accounting (AAA) is not supported.
- The IPSec idle timer is configured per IPSec profile using the ipsec-profile command. The idle time is the same for all VPN-SIP sessions that use a specific IPSec profile.
- Track objects that are used for IPSLA monitoring, have a maximum limit of 1000 objects in Cisco IOS software. When one track object is used to track one peer router, maximum number of VPN-SIP sessions that one IOS device can have is limited by the maximum number of track objects.
- Only one local SIP number is supported on Cisco IOS software.
- If there is a pre-existing statically configured policy, the programmatically configured QoS policy is not applied and session fails. Remove any statically configured QoS policy on the SVTI interface.
- On all Cisco ISR 1100 series routers, the supported scale of VPN-SIP feature is 300 sessions.
- Cisco does not support the interoperability with VPN-SIP implementation of other vendors.
- For the class policies included in the `policy-map` attached to the VPN-SIP tunnel, only Priority Queueing and Class-Based Weighted Fair Queueing (CBWFQ) are supported.
- For CBWFQ configurations, only the `bandwidth percent` *`percent`* command is supported. The `bandwidth` *`bandwidth`* command is not supported as the bandwidth of the VPN-SIP session varies depending on the negotiation with the peer router.

How to Configure VPN-SIP

Configuring VPN-SIP

The following steps describe the process of configuring VPN-SIP:

1. Configure the tunnel authentication using third party certificates, self-signed certificates, or pre-shared keys.

a. Tunnel Authentication using Certificates

Configure a trustpoint to obtain a certificate from a certification authority (CA) server that is located in the customer's network. This is required for tunnel authentication. Use the following configuration:

```
peer1(config)# crypto pki trustpoint CA
 enrollment url http://10.45.18.132/
 serial-number none
 subject-name CN=peer2
 revocation-check crl
 rsakeypair peer2

peer2(config)# crypto pki authenticate CA
Certificate has the following attributes:
       Fingerprint MD5: F38A9B4C 2D80490C F8E7581B BABE7CBD
      Fingerprint SHA1: 4907CC36 B1957258 5DFE23B2 649E7DDA 99BDB7C3
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.

peer2(config)#crypto pki enroll CA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.
Password:
Re-enter password:
% The subject name in the certificate will include: CN=peer2
% The subject name in the certificate will include: peer2
% Include an IP address in the subject name? [no]:
Request certificate from CA? [yes/no]: yes
% Certificate request sent to Certificate Authority
% The 'show crypto pki certificate verbose CA' command will show the fingerprint.
Certificate map for Trustpoint
crypto pki certificate map data 1
issuer-name co cn = orange
```

b. Tunnel authentication using self-signed certificate

Configure a PKI trust point to generate a self-signed certificate on the device, when authenticating using a self-signed certificate. Use the following configuration:

```
peer4(config)#crypto pki trustpoint Self
    enrollment selfsigned
    revocation-check none
    rsakeypair myRSA
    exit
crypto pki enroll Self

Do you want to continue generating a new Self Signed Certificate? [yes/no]: yes
% Include the router serial number in the subject name? [yes/no]: yes
% Include an IP address in the subject name? [no]: no
Generate Self Signed Router Certificate? [yes/no]: yes

Router Self Signed Certificate successfully created
```

c. Configure tunnel authentication using a pre-shared key

```
crypto ikev2 keyring keys
peer peer1
identity key-id 1234
pre-shared-key key123
```

2. • Configure IKEv2 Profile for Certificate

```
crypto ikev2 profile IPROF
match certificate data
identity local key-id 5678
authentication remote rsa-sig
authentication local rsa-sig
keyring local keys
pki trustpoint self
nat force-encap
```

• Configure an IKEv2 Profile for pre-shared keys

```
crypto ikev2 profile IPROF
match identity remote any
identity local key-id 5678
authentication remote pre-share
authentication local pre-share
keyring local keys
nat force-encap
```

Note To complete the IKEv2 SA configuration, the **nat force-encap** command must be configured on both peers. Since, UDP encapsulation is negotiated in SDP, IKEv2 must start and continue on port 4500.

3. Configure an IPsec profile

```
crypto ipsec profile IPROF
set security-association idle-time 2000
```

4. Configure a LAN side interface

```
interface Vlan101
         ip address 192.0.2.3 255.255.255.0
         no shutdown
!
     interface GigabitEthernet2
         switchport access vlan 101
         no ip address
```

5. Configure a loopback interface

The loopback interface is used as the source interface for the secondary VPN tunnel.

```
interface loopback 1
    ip address 192.0.2.1 255.0.0.0
    ip nat inside
```

6. Configure a secondary interface.

Note Make sure the secondary interface is configured to receive the IP address, SIP server address, and vendor specific information via DHCP.

```
interface GigabitEthernet8
     ip dhcp client request sip-server-address
     ip dhcp client request vendor-identifying-specific
     ip address dhcp
    ip nat outside
```

7. Configure the tunnel interface

```
interface Tunnel1
     ip address 192.0.2.1 255.255.255.255
     load-interval 30
     tunnel source Loopback1
     tunnel mode ipsec ipv4
     tunnel destination dynamic
     tunnel protection ipsec profile IPROF ikev2-profile IPROF
     vpn-sip local-number 5678 remote-number 1234 bandwidth 1000
```

Use the **vpn-sip local-number** *local-number* **remote-number** *remote-number* **bandwidth** *bw-number* command to configure the sVTI interface for VPN-SIP. Bandwidth is the maximum data transmission rate that must be negotiated with this peer and the negotiated value is set on the tunnel interface. Allowed values are 64, 128, 256, 512, and 1000 kbps.

Once an SVTI is configured for VPN-SIP, changes cannot be made to tunnel mode, tunnel destination, tunnel source, and tunnel protection. To change the mode, source, destination, or tunnel protection you must remove the VPN-SIP configuration from the SVTI interface.

8. Add static routes to destination networks

Add a secondary route with a higher metric.

```
ip route 192.0.2.168 255.255.255.0 Tunnel0 track 1
ip route 192.0.2.168 255.255.255.0 Tunnel1 254
```

9. Configure IP SLA

```
ip sla 1
         icmp-echo 192.0.2.11
         threshold 500
         timeout 500
         frequency 2
        ip sla schedule 1 life forever start-time now
```

10. Configure route tracking

```
track 1 ip sla 1 reachability
```

11. Enable VPN-SIP

```
vpn-sip enable
vpn-sip local-number 5678 address ipv4 GigabitEthernet8
vpn-sip tunnel source Loopback1
vpn-sip logging
```

To configure VPN-SIP, you must configure local SIP number and local address. The **vpn-sip local-number** *SIP-number* **address ipv4** *WAN-interface-name* command configures the local SIP number that is used for SIP call and the associated IPv4 address.

Note Only IPv4 addresses can be configured. Crypto module does not support dual stack.

- Backup WAN interface address may change based on DHCP assignment.

When the primary WAN interface is functional, the destination of the VPN-SIP tunnel is set to the backup WAN interface, so that the tunnel interface is active. Destination is set to IP address of the peer that is learnt from SDP of SIP negotiation when traffic is routed to the tunnel interface. When primary WAN interface fails and the back routes are activated, packets are routed to the sVTI through backup.

Note We recommend that you use an unused non-routable address as the address of the loopback interface and do not configure this loopback interface for any other purpose. Once a loopback interface is configured, VPN-SIP listens to any updates to the interface and blocks them. The **vpn-sip logging** command enables the system logging of VPN-SIP module for events, such as session up, down, or failure.

Verifying VPN-SIP on a Local Router

Verifying Registration Status

```
Peer1# show vpn-sip registration-status
 SIP registration of local number 0388881001 : registered 10.6.6.50
```

Verifying SIP Registrar

```
Peer1#show vpn-sip sip registrar

Line          destination      expires(sec)  contact   transport     call-id
=============================================================================================
0388881001    example.com         2359        10.6.6.50  UDP
3176F988-9EAA11E7-8002AFA0-8EF41435
```

Verifying VPN-SIP Status

```
Peer1#show vpn-sip session detail
VPN-SIP session current status

Interface: Tunnel1
   Session status: SESSION_UP (I)
   Uptime        : 00:00:42
   Remote number : 0388881001 =====> This is the Remote Router's SIP number
   Local number  : 0388882001 =====> Local router's SIP number
   Remote address:port: 10.6.6.49:50002
   Local address:port : 10.6.6.50:50001
   Crypto conn handle: 0x8000017D
   SIP Handle        : 0x800000C7
   SIP callID        : 1554
   Configured/Negotiated bandwidth: 64/64 kbps
```

Verifying Crypto Session

```
Peer1# show crypto session detail
Crypto session current status
Code: C - IKE Configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
X - IKE Extended Authentication, F - IKE Fragmentation
R - IKE Auto Reconnect, U - IKE Dynamic Route Update
S - SIP Vpn-sip

Interface: Tunnel1
Profile: IPROF
Uptime: 00:03:53
Session status: UP-ACTIVE
Peer: 10.6.6.49 port 4500 fvrf: (none) ivrf: (none)
      Phase1_id: 10.6.6.49
```

```
     Desc: (none)
  Session ID: 43
  IKEv2 SA: local 10.11.1.1/4500 remote 10.6.6.49/50002 Active
          Capabilities:S connid:1 lifetime:23:56:07 ====> Capabilities:S indicates this is
a  SIP VPN_SIP Session
  IPSEC FLOW: permit ip 0.0.0.0/0.0.0.0 0.0.0.0/0.0.0.0
        Active SAs: 2, origin: crypto map
        Inbound:  #pkts dec'ed 6 drop 0 life (KB/Sec) 4222536/3366
        Outbound: #pkts enc'ed 4 drop 0 life (KB/Sec) 4222537/3366
```

Verifying IP NAT Translations

```
Peer1#sh ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
udp 2.2.2.2:4500       10.6.6.50:50001    10.6.6.49:50002    10.6.6.49:50002
```

Verifying DHCP SIP Configuration

```
Peer9#show vpn-sip sip dhcp
SIP DHCP Info

SIP-DHCP interface:  GigabitEthernet8

SIP server address:
Domain name:         dns:example.com
```

Verifying VPN-SIP on a Remote Router

Verifying VPN-SIP Registration Status on a Remote Router

```
Peer2# show vpn-sip registration-status
 SIP registration of local number 0388882001 : registered 10.6.6.49
```

Verifying VPN-SIP Registrar on a Remote Router

```
Peer2# show vpn-sip sip registrar
Line          destination      expires(sec)  contact    transport    call-id
=============================================================================================
0388882001    example.com    2478         10.6.6.49   UDP
E6F23809-9EAB11E7-80029279-40B97F59
```

Verifying VPN-SIP Session Details on a Remote Router

```
Peer2# show vpn-sip session detail
VPN-SIP session current status
Interface: Tunnel1
   Session status: SESSION_UP (R)
   Uptime        : 00:00:21
   Remote number : 0388882001 =====> This is the Peer1 Router's SIP number
   Local number  : 0388881001 =====> Local router's SIP number
   Remote address:port: 10.6.6.50:50001
   Local address:port : 10.6.6.49:50002
   Crypto conn handle: 0x8000017E
   SIP Handle        : 0x800000BE
   SIP callID        : 1556
   Configured/Negotiated bandwidth: 1000/64 kbps
```

Verifying Crypto Session Details on a Remote Router

```
Peer2 #show crypto session detail
Crypto session current status
Code: C - IKE Configuration mode, D - Dead Peer Detection
K - Keepalives, N - NAT-traversal, T - cTCP encapsulation
X - IKE Extended Authentication, F - IKE Fragmentation
R - IKE Auto Reconnect, U - IKE Dynamic Route Update
S - SIP VPN-SIP

Interface: Tunnel1
Profile: IPROF
Uptime: 00:02:32
Session status: UP-ACTIVE
Peer: 10.6.6.50 port 50001 fvrf: (none) ivrf: (none)
      Phase1_id: 10.6.6.50
      Desc: (none)
  Session ID: 147
  IKEv2 SA: local 10.17.1.1/4500 remote 10.6.6.50/50001 Active
          Capabilities:S connid:1 lifetime:23:57:28 ====> Capabilities:S indicates this is
 a  SIP VPN-SIP Session
  IPSEC FLOW: permit ip 0.0.0.0/0.0.0.0 0.0.0.0/0.0.0.0
        Active SAs: 2, origin: crypto map
        Inbound:  #pkts dec'ed 4 drop 0 life (KB/Sec) 4293728/3448
        Outbound: #pkts enc'ed 6 drop 0 life (KB/Sec) 4293728/3448
```

Verifying IP NAT Translations on a Remote Router

```
Peer2#show ip nat translations
Pro Inside global      Inside local       Outside local      Outside global
udp 3.3.3.3:4500       10.6.6.49:50002    10.6.6.50:50001    10.6.6.50:50001
```

Configuring QoS for VPN-SIP

Optionally, you can apply a quality of service (QoS) policy to the VPN-SIP. A QoS policy provides secure, predictable, measurable, and sometimes guaranteed services to certain types of traffic.

1. Configure the appropriate policy map.

```
Device(config)#class-map match-all UDP
 match protocol ip
!
policy-map CBWFQ
 class UDP
  bandwidth percent 60
  queue-limit 12 packets
```

2. Attach the policy-map to the VPN-SIP:

```
Device(config)#interface Tunnel1
.
.
.
vpn-sip local-number 5678 remote-number 1234 bandwidth 1000 service-policy CBWFQ
```

Note When the VPN-SIP session is successfully negotiated and comes up, an implicit service policy is automatically attached to the tunnel interface. If you run the `show running-config` command for this interface, the implicit service policy is not displayed. Any `policy-map` that you create on the device becomes a child policy of this implicit service policy.

Verifying QoS for VPN-SIP

Verifying the Application of the Policy Map

```
Peer1#sh policy-map int tun1
 Tunnel1

  Service-policy output: VPN-SIP-Tunnel1-Bandwidth

    Class-map: class-default (match-any)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0
      QoS Set
        dscp cs4
          Packets marked 0
      shape (average) cir 1000000, bc 4000, be 4000
      target shape rate 1000000

      Service-policy : CBWFQ

        Class-map: UDP (match-all)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: protocol ip
          Queueing
          queue limit 12 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0
          bandwidth 60% (600 kbps)


        Class-map: class-default (match-any)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: any

          queue limit 64 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0
Peer1#sh vpn-sip session detail
VPN-SIP session current status

Interface: Tunnel1
   Session status: SESSION_UP (R)
   Uptime        : 00:00:15
   Remote number : 5678
   Local number  : 1234
   Remote address:port: 6.6.6.40:51878
   Local address:port : 6.6.6.89:50010
   Crypto conn handle: 0x40000017
   SIP Handle        : 0x4000000B
   SIP callID        : 2288
   Configured/Negotiated bandwidth: 1000/1000 kbps
   Applied service policy: CBWFQ
```

Verifying the Flow of Traffic

After sending UDP traffic in the direction of the policy, verify the flow of traffic as follows:

```
Peer1#sh policy-map int tun1
 Tunnel1

  Service-policy output: VPN-SIP-Tunnel1-Bandwidth

    Class-map: class-default (match-any)
      105782 packets, 4865972 bytes
      5 minute offered rate 130000 bps, drop rate 0000 bps
      Match: any
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/98707/0
      (pkts output/bytes output) 7068/890568
      QoS Set
        dscp cs4
          Packets marked 105782
      shape (average) cir 1000000, bc 4000, be 4000
      target shape rate 1000000

      Service-policy : CBWFQ

        Class-map: UDP (match-all)
          105775 packets, 4865650 bytes
          5 minute offered rate 130000 bps, drop rate 331000 bps
          Match: protocol ip
          Queueing
          queue limit 12 packets
          (queue depth/total drops/no-buffer drops) 11/98707/0
          (pkts output/bytes output) 7068/890568
          bandwidth 60% (600 kbps)


        Class-map: class-default (match-any)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: any

          queue limit 64 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0
```

Configuration Examples for VPN-SIP

Using self-signed certificates for authentication

The following is sample configuration to configure VPN-SIP using self-signed certificates for authentication. There is no distinction between initiator and responder role in VPN-SIP. The configuration on a peer node will be identical with local SIP numbers changed.

```
// Self-signed certificate
crypto pki trustpoint selfCert
  rsakeypair myRSA
  enrollment selfsigned
  revocation-check none
!
crypto ikev2 profile vpn-sip-profile
```

```
 match identity remote any
 authentication local rsa-sig
 authentication remote rsa-sig
 pki trustpoint selfCert // Use same self-signed trustpoint for sign and verify
 nat force-encap
!
crypto ipsec profile vpn-sip-ipsec
 set security-association idle-time 120
!
vpn-sip enable
vpn-sip local-number 0388883001 address ipv4 GigabitEthernet1
vpn-sip tunnel source Loopback11
vpn-sip logging
!
// one tunnel per peer – configuration is for peer with a SIP-number of 0388884001
int tunnel0
 ip unnumbered loopback 0
 tunnel source loopback11
 tunnel mode ipsec ipv4
 tunnel destination dynamic
 tunnel protection ipsec profile vpn-sip-ipsec ikev2-profile vpn-sip-profile
 vpn-sip local-number 0388883001 remote-number 0388884001 bandwidth 1000
!
// ip unnumbered of tunnel interfaces
int loopback 0
  ip address 10.21.1.1 255.255.255.255
!
int loopback11
ip address 10.9.9.9 255.255.255.255
ip nat inside
!
// one tunnel per peer – this is for peer with SIP-number 0388885001
int tunnel1
 ip unnumbered loopback 0
 tunnel source loopback11
 tunnel mode ipsec ipv4
 tunnel destination dynamic
 tunnel protection ipsec profile vpn-sip-ipsec ikev2-profile iprof
 vpn-sip sip-local 0388883001 sip-remote 0388885001 bandwidth 1000
!
interface GigabitEthernet8
 ip dhcp client request sip-server-address
 ip dhcp client request vendor-identifying-specific
 ip address dhcp
ip nat outside


// backup routes configured with higher AD so that these routes will be activated only when
 primary path goes down. AD need to be chosen to be greater than that of primary route.
ip route 10.0.0.0 255.0.0.0 tunnel 0 250
ip route 10.1.0.0 255.0.0.0 tunnel 0 250
ip route 10.2.0.0 255.0.0.0 tunnel 0 250
ip route 10.3.0.0 255.0.0.0 tunnel 0 250
```

Troubleshooting for VPN-SIP

Viewing Tunnel Interface in Show Output

Symptom

Show VPN-SIP session doesn't show any information about the tunnel interface. In the following example, information about the tunnel interface, tunnel1 is not shown:

```
Peer5-F#show vpn-sip session
VPN-SIP session current status

Interface: Tunnel2
   Session status: READY_TO_CONNECT
   Remote number : 0334563333
   Local number  : 0623458888
   Remote address:port: 10.10.0.0:0
   Local address:port : 192.0.2.22:0

Interface: Tunnel3
   Session status: READY_TO_CONNECT
   Remote number : 0323452222
   Local number  : 0623458888
   Remote address:port: 10.10.0.0:0
   Local address:port : 192.0.2.22:0

Interface: Tunnel4
   Session status: READY_TO_CONNECT
   Remote number : 0612349999
   Local number  : 0623458888
   Remote address:port: 10.10.0.0:0
   Local address:port : 192.0.2.22:0

Interface: Tunnel6
   Session status: READY_TO_CONNECT
   Remote number : 0634567777
   Local number  : 0623458888
   Remote address:port: 10.10.0.0:0
   Local address:port : 172.30.18.22:0
```

Possible Cause

VPN-SIP is not configured on the tunnel interface

```
Peer5-F#sh run int tun1
Building configuration...

Current configuration : 201 bytes
!
interface Tunnel1
 ip address 10.5.5.5 255.255.255.0
 tunnel source Loopback11
 tunnel mode ipsec ipv4
 tunnel destination dynamic
 tunnel protection ipsec profile test-prof ikev2-profile test
end
```

Recommended Action

Configure VPN-SIP on the tunnel interface.

```
:

Peer5-F#show running interface tunnel 1
Building configuration...

Current configuration : 278 bytes
!
interface Tunnel1
 ip address 10.5.5.5 255.255.255.255
 tunnel source Loopback11
```

```
 tunnel mode ipsec ipv4
 tunnel destination dynamic
 tunnel protection ipsec profile test-prof ikev2-profile test
 vpn-sip local-number 0623458888 remote-number 0312341111 bandwidth 1000
end
```

Following is the running output for the above scenario:

```
Peer5-F#show vpn-sip session detail
VPN-SIP session current status

Interface: Tunnel1
   Session status: READY_TO_CONNECT
   Remote number : 0312341111
   Local number  : 0623458888
   Remote address:port: 10.0.0.0:0
   Local address:port : 172.30.18.22:0

   Crypto conn handle: 0x8000002C
   SIP Handle        : 0x0
   SIP callID        : --
   Configured/Negotiated bandwidth: 1000/0 kbps

Interface: Tunnel2
   Session status: READY_TO_CONNECT
   Remote number : 0334563333
   Local number  : 0623458888
   Remote address:port: 10.0.0.0:0
   Local address:port : 172.30.18.22:0
   Crypto conn handle: 0x80000012
   SIP Handle        : 0x0
   SIP callID        : --
   Configured/Negotiated bandwidth: 512/0 kbps

Interface: Tunnel3
   Session status: READY_TO_CONNECT
   Remote number : 0323452222
   Local number  : 0623458888
   Remote address:port: 10.0.0.0:0
   Local address:port : 172.30.18.22:0
   Crypto conn handle: 0x80000031
   SIP Handle        : 0x0
   SIP callID        : --
   Configured/Negotiated bandwidth: 512/0 kbps

Interface: Tunnel4
   Session status: READY_TO_CONNECT
   Remote number : 0612349999
   Local number  : 0623458888
   Remote address:port: 10.0.0.0:0
   Local address:port : 172.30.18.22:0
   Crypto conn handle: 0x8000002F
   SIP Handle        : 0x0
   SIP callID        : --
   Configured/Negotiated bandwidth: 1000/0 kbps

Interface: Tunnel6
   Session status: READY_TO_CONNECT
   Remote number : 0634567777
   Local number  : 0623458888
   Remote address:port: 10.0.0.0:0
   Local address:port : 172.30.18.22:0
   Crypto conn handle: 0x80000026
   SIP Handle        : 0x0
```

```
   SIP callID        : --
   Configured/Negotiated bandwidth: 1000/0 kbps
```

Troubleshooting SIP Registration Status

Symptom

SIP registration status is Not Registered

```
Peer5#show vpn-sip sip registrar
Line          destination      expires(sec)  contact
transport     call-id
============================================================

Peer5-F#show vpn-sip registration-status

 SIP registration of local number 0623458888 : not registered
```

Possible Cause

IP address is not configured on the WAN interface.

```
Peer5#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         unassigned      YES unset  down                  down
GigabitEthernet0/1         unassigned      YES unset  up                    up
GigabitEthernet0/2         unassigned      YES unset  down                  down
GigabitEthernet0/3         unassigned      YES unset  down                  down
GigabitEthernet0/4         unassigned      YES unset  up                    up
GigabitEthernet0/5         10.5.5.5        YES manual up                    up
Vlan1                      10.45.1.5       YES NVRAM  up                    up
NVI0                       10.1.1.1        YES unset  up                    up
Loopback1                  10.1.1.1        YES NVRAM  up                    up
Loopback5                  10.5.5.5        YES NVRAM  administratively down down
Loopback11                 10.11.11.11     YES NVRAM  up                    up
Tunnel1                    10.5.5.5        YES NVRAM  up                    down
Tunnel2                    10.2.2.2        YES NVRAM  up                    down
Tunnel3                    10.3.3.3        YES NVRAM  up                    down
Tunnel4                    10.4.4.4        YES NVRAM  up                    down
Tunnel6                    10.8.8.8        YES NVRAM  up                    down

Peer5-F#show run interface gigabitEthernet 0/4
Building configuration...

Current configuration : 213 bytes
!
interface GigabitEthernet0/4
 ip dhcp client request sip-server-address
 ip dhcp client request vendor-identifying-specific
 no ip address           ====> no IP address
 ip nat outside
 ip virtual-reassembly in
 duplex auto
 speed auto
end
```

Recommended Action

Use the **ip address dhcp** command to configure the interface IP address.

```
Peer5-F#show running-config interface gigabitEthernet 0/4
Building configuration...

Current configuration : 215 bytes
```

```
!
interface GigabitEthernet0/4
 ip dhcp client request sip-server-address
 ip dhcp client request vendor-identifying-specific
 ip address dhcp         ====> configure IP address DHCP
 ip nat outside
 ip virtual-reassembly in
 duplex auto
 speed auto
end


Peer5-F#show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         unassigned      YES unset  down                  down
GigabitEthernet0/1         unassigned      YES unset  up                    up
GigabitEthernet0/2         unassigned      YES unset  down                  down
GigabitEthernet0/3         unassigned      YES unset  down                  down
GigabitEthernet0/4         172.30.18.22    YES DHCP   up                    up
GigabitEthernet0/5         10.5.5.5        YES manual up                    up
Vlan1                      10.45.1.5       YES NVRAM  up                    up
NVI0                       10.1.1.1        YES unset  up                    up
Loopback1                  10.1.1.1        YES NVRAM  up                    up
Loopback5                  10.5.5.5        YES NVRAM  administratively down down
Loopback11                 10.11.11.11     YES NVRAM  up                    up
Tunnel1                    10.6.5.5        YES NVRAM  up                    down
Tunnel2                    10.2.2.2        YES NVRAM  up                    down
Tunnel3                    10.3.3.3        YES NVRAM  up                    down
Tunnel4                    10.4.4.4        YES NVRAM  up                    down
Tunnel6                    10.8.8.8        YES NVRAM  up                    down

Peer5-F#show vpn-sip sip registrar
Line          destination      expires(sec)  contact
transport     call-id
============================================================
0623458888    example.com   2863          172.30.18.22
UDP           1E83ECF0-AF0611E7-802B8FCF-594EB9E7@10.50.18.22

Peer5-F#show vpn-sip registration-status

 SIP registration of local number 0623458888 : registered 172.30.18.22
```

Session stuck in Negotiating IKE state

Symptom

VPN-SIP session stuck in Negotiating IKE state.

```
Peer5#show vpn-sip session remote-number 0612349999 detail
VPN-SIP session current status

Interface: Tunnel4
   Session status: NEGOTIATING_IKE (R)
   Uptime        : 00:00:58
   Remote number : 0612349999
   Local number  : 0623458888
   Remote address:port: 72.30.168.3:24825
   Local address:port : 72.30.168.22:50012
   Crypto conn handle: 0x8000002E
   SIP Handle        : 0x8000000C
   SIP callID        : 16
   Configured/Negotiated bandwidth: 1000/1000 kbps
```

Possible Cause

Bad configuration related to IKEv2.

In the following example the Key ID that is configured in the keyring does not match the SIP numberof the remote peer.

```
Peer5-F#show running-config interface tunnel 4
Building configuration...

Current configuration : 276 bytes
!
interface Tunnel4
 ip address 10.4.4.4 255.255.255.0
 tunnel source Loopback11
 tunnel mode ipsec ipv4
 tunnel destination dynamic
 tunnel protection ipsec profile test-prof ikev2-profile test
 VPN-SIP local-number 0623458888 remote-number 0612349999 bandwidth 1000  ====> Remote
number mentioned here doesn’t match the remote number in the keyring
end

IKEv2 Keyring configs:
!
crypto ikev2 keyring keys
 peer peer1
  identity key-id 0312341111
  pre-shared-key psk1
 !
 peer abc
  identity key-id 0345674444
  pre-shared-key psk1
 !
 peer peer2
  identity key-id 0334563333
  pre-shared-key psk10337101690
 !
 peer peer6
  identity key-id 0634567777
  pre-shared-key cisco123
 !
 peer peer3
  identity key-id 0323452222
  pre-shared-key cisco123
 !
 peer peer4
  identity key-id 0645676666
  pre-shared-key psk1
 !
 peer NONID
  identity fqdn example.com
  pre-shared-key psk1
 !
!
!
crypto ikev2 profile test
 match identity remote any
 identity local key-id 0623458888
 authentication remote pre-share
 authentication local pre-share
 keyring local keys
 dpd 10 6 periodic
 nat force-encap
```

Recommended Action

Correct the keyring configurations.

```
rypto ikev2 keyring keys
 peer peer1
  identity key-id 0312341111
  pre-shared-key psk1
 !
 peer abc
  identity key-id 0345674444
  pre-shared-key psk1
 !
 peer peer2
  identity key-id 0334563333
  pre-shared-key psk1
 !
 peer peer6
  identity key-id 0634567777
  pre-shared-key psk1
 !
 peer peer3
  identity key-id 0323452222
  pre-shared-key psk1
 !
 peer peer4
  identity key-id 0612349999
  pre-shared-key psk1
 !
 peer NONID
  identity fqdn example.com
  pre-shared-key psk1
 !
!
!
crypto ikev2 profile test
 match identity remote any
 identity local key-id 0623458888
 authentication remote pre-share
 authentication local pre-share
 keyring local keys
 dpd 10 6 periodic
 nat force-encap
!

Peer5-F#show vpn-sip session remote-number  0612349999 detail
VPN-SIP session current status

Interface: Tunnel4
   Session status: SESSION_UP (R)
   Uptime        : 00:02:04
   Remote number : 0612349999
   Local number  : 0623458888
   Remote address:port: 198.51.100.3:24845
   Local address:port : 198.51.100.22:50020
   Crypto conn handle: 0x8000004E
   SIP Handle        : 0x80000014
   SIP callID        : 24
   Configured/Negotiated bandwidth: 1000/1000 kbps
```

Troubleshooting Session Initiation

Symptom

Session does not initiate and gets stuck in Negotiating IKE state

Possible Cause

Fagmentation of IKE packets when a large PKI certificate is included in the IKE authentication message.

Recommended Action

Configure IKEv2 fragmentation on the routers.

Debug Commands

The folllwing debug commands are available to debug VPN-SIP configuration:

Table 4: debug commands

Command Name	Description
debug vpn-sip event	Prints debug messages for SVTI registration with VPN-SIP, SIP registration, call setup, and so on.
debug vpn-sip errors	Prints error messages only when an error occurs during initialization, registration, call setup, and so on.
debug vpn-sip sip all	Enables all SIP debugging traces.
debug vpn-sip sip calls	Enables SIP SPI calls debugging trace.
debug vpn-sip sip dhcp	Enables SIP-DHCP debugging trace
debug vpn-sip sip error	Enables SIP error debugging trace
debug vpn-sip sip events	Enables SIP events debugging trace.
debug vpn-sip sip feature	Enables feature level debugging.
debug vpn-sip sip function	Enables SIP function debugging trace.
debug vpn-sip sip info	Enables SIP information debugging trace.
debug vpn-sip sip level	Enables information level debugging.
debug vpn-sip sip media	Enables SIP media debugging trace.
debug vpn-sip sip messages	Enables SIP SPI messages debugging trace
debug vpn-sip sip non-call	Enables Non-Call-Context trace (OPTIONS, SUBSCRIBE, and so on)
debug vpn-sip sip preauth	Enable SIP preauth debugging trace.
debug vpn-sip sip states	Enable SIP SPI states debugging trace.
debug vpn-sip sip translate	Enables SIP translation debugging trace.
debug vpn-sip sip transport	Enables SIP transport debugging traces.
debug vpn-sip sip verbose	Enables verbose mode.

Additional References for VPN-SIP

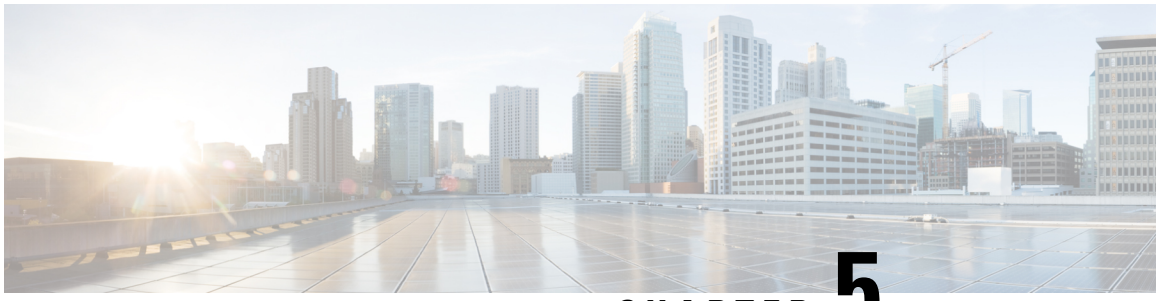
Standards and RFCs

Standard/RFC	Title
RFC 6193 (with Restrictions)	Media Description for the Internet Key Exchange Protocol (IKE) in the Session Description Protocol (SDP)

Feature Information for VPN-SIP

Table 5: Feature Information for VPN-SIP

Feature Name	Releases	Feature Information
Session Initiation Protocol Triggered VPN		VPN-SIP is a service offered by service providers where a VPN is setup for on-demand media or application sharing between peers, using Session Initiation Protocol (SIP). The following commands were introduced: **nat force-encap, show vpn-sip session, show vpn-sip sip, show vpn-sip registration-status, vpn-sip local-number, vpn-sip logging, vpn-sip tunnel source**.



CHAPTER 5

Deleting Crypto Sessions of Revoked Peer Certificates

The Delete Crypto Sessions of Revoked Peer Certificates on CRL Download feature deletes an active crypto session with a peer if its certificate is found to be revoked when downloading a new CRL.

Finding Feature Information

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

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

Restrictions for Deleting Crypto Sessions of Revoked Peer Certificates

- If revocation check is turned off and this feature is enabled, the IKE database is not populated with the number of sessions. The show outputs do not display information about the deleted sessions.
- Frequent enabling and disabling of this feature (with active sessions on the device) is not recommended.
- Frequent CRL downloads (in a span of 30 minutes) for the same issuername (CA server) is not recommended.

- CRL cache must be enabled. CRL caching cannot be disabled for trustpoint-based prefetch. However, it is possible to disable CRL caching for URL-based prefetch.
- In case of autoenrollment on IKE, the sessions are not deleted until the next IKE rekey, whereas in case of IKEv2, the tunnel must be cleared manually or wait until the certificate expires.
- If IKE has database of "issuer-name" and "SN" populated and receives a notification from PKI about certificate revocation, IKE would act on the PKI notification.

Information About Deleting Crypto Sessions of Revoked Peer Certificates

How a Crypto Session is Deleted

1. When negotiating via certificate authentication, the peer sends the CERT payload to the device, whcih parses each certificate to store information about serial number and the issuer names. This information forms the list of serial numbers issued by the corresponding CA server and is passed to PKI for revocation check.
2. If the revocation-check crl command is configured for a trustpoint, PKI informs IKE about the revocation check thereby disabling IKE from unnecessarily storing unwanted peer certification information.
3. After a successful CRL download, PKI sends IKE a notification, which contains the "issuer-name." The CRL signature and content is verified. If there is no change in CRL content, PKI does not notify IKE.
4. If PKI notifies IKE containing the issuer name, IKE prepares a list of serial numbers for an issuer name and passes this list to PKI to verify if the serial numbers in the list are revoked.
5. PKI performs revocation check on the serial number list received from the IKE and checks the list against the downloaded CRL. The revoked serial number list is returned to IKE.
6. On a notification from PKI containing the list of revoked serial numbers, IKE identifies and deletes sessions pertaining to those serial numbers those sessions.

How to Enable Deletion of Crypto Sessions for Revoked Peer Certificates

Enabling Deletion of Crypto Sessions

Perform this task to enable the deletion of crypto sessions for revoked certificates.

SUMMARY STEPS

1. **enable**
2. **clear crypto session**
3. **configure terminal**

4. Do one of the following:
   - **crypto isakmp disconnect-revoked-peers**
   - **crypto ikev2 disconnect-revoked-peers**
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**clear crypto session** **Example:** `Device# clear crypto session`	(Optional) Deletes IPsec crypto sessions and IKE and security associations. **Note** Use this command to enable the feature for previously established sessions, else the feature is enabled for new sessions only.
Step 3	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.

Step 4	Do one of the following:
	• **crypto isakmp disconnect-revoked-peers**
	• **crypto ikev2 disconnect-revoked-peers**
	Example:
	`Device(config)# crypto isakmp disconnect-revoked-peers`
	Example:
	`Device(config)# crypto ikev2 disconnect-revoked-peers`
		Disconnects IKE or IKEv2 crypto sessions with peers having revoked certificates.
		For this command to take effect, reconnected the existing sessions.

Step 5	**end** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Verifying the Delete Crypto Session Capability for a Revoked Peer Certificate

Perform this task to verify if the delete crypto session capability is displayed in the show output.

SUMMARY STEPS

1. **enable**
2. **show crypto isakmp peers**

3. **show crypto ikev2 session detail**

DETAILED STEPS

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show crypto isakmp peers**

Example:

```
Device# show crypto isakmp peers
```

Displays Internet Security Association and Key Management Protocol (ISAKMP) peer descriptions.

Step 3 **show crypto ikev2 session detail**

Example:

```
Device# show crypto ikev2 session detail
```

Displays the status of active Internet Key Exchange Version 2 (IKEv2) sessions.

Configuration Examples for Deleting Crypto Sessions of Revoked Peer Certificates

Example: Enabling Deletion of Crypto Sessions for an IKE Session

```
Device> enable
Device# clear crypto session
Device# configure terminal
Device(config)# crypto isakmp disconnect-revoked-peers
Device# show crypto isakmp peers

Peer: 150.1.1.2 Port: 500 Local: 150.1.1.1
 Phase1 id: 150.1.1.2
 Disconnect Revoked Peer: Enabled
```

Example: Enabling Deletion of Crypto Sessions for an IKEv2 Session

```
Device> enable
Device# clear crypto session
Device# configure terminal
Device(config)# crypto ikev2 disconnect-revoked-peers
Device# show crypto ikev2 session detail
```

```
Session-id:1, Status:UP-ACTIVE, IKE count:1, CHILD count:1
Tunnel-id   Local          Remote           fvrf/ivrf          Status
1           10.0.0.1/500    10.0.0.2/500      (none)/(none)      READY
      Encr: 3DES, Hash: SHA96, DH Grp:2, Auth: PSK
      Life/Remaining/Active Time: 86400/86157/248 sec
      CE id: 0, Session-id: 1, MIB-id: 1
      Status Description: Negotiation done
      Local spi: 750CBE827434A245       Remote spi: 4353FEDBABEBF24C
      Local id:          10.0.0.1        Remote id:          10.0.0.2
      Local req mess id:    0           Remote req mess id: 0
      Local next mess id:    0          Remote next mess id: 2
      Local req queued:    0            Remote req queued: 0
      Local window:    5                Remote window: 5
      DPD configured for 0 seconds
      NAT-T is not detected
      Disconnect Revoked Peer: Enabled
Child sa: local selector  10.0.0.1/0 - 10.0.0.1/65535
          remote selector 10.0.0.2/0 - 10.0.0.2/65535
          ESP spi in/out: 0x9360A95/0x6C340600
          CPI in/out: 0x9FE5/0xC776
          AH spi in/out: 0x0/0x0
          Encr: AES CBC, keysize: 128, esp_hmac: SHA96
          ah_hmac: Unknown - 0, comp: IPCOMP_LZS, mode tunnel
```

Additional References for Deleting Crypto Sessions of Revoked Peers

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*
Configuring IKE	*Configuring Internet Key Exchange for IPsec VPNs*
Configuring IKEv2	*Configuring Internet Key Exchange Version 2 and FlexVPN Site-to-Site*
Recommended cryptographic algorithms	*Next Generation Encryption*

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

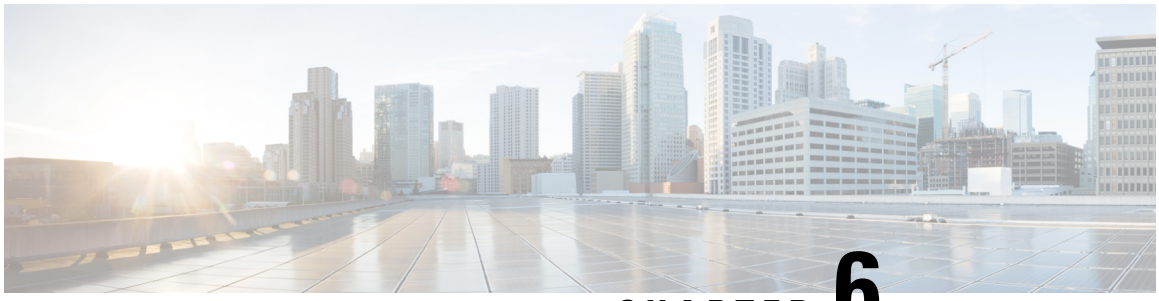
Feature Information for Deleting Crypto Sessions of Revoked Peer Certificates

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

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

Table 6: Feature Information for Deleting Crypto Sessions of Revoked Peer Certificates

Feature Name	Releases	Feature Information
Delete crypto session(s) of revoked peer cert(s) on CRL download		The Delete Crypto Sessions of Revoked Peer Certificates on CRL Download feature deletes an active crypto session with a peer if its certificate is found to be revoked when downloading a new CRL. The following commands were introduced or modified: **crypto ikev2 disconnect-revoked-peers**, **crypto isakmp disconnect-revoked-peers**, **show crypto isakmp peers**, **show crypto ikev2 session detail**.



CHAPTER 6

Crypto Conditional Debug Support

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

Finding Feature Information

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

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

Prerequisites for Crypto Conditional Debug Support

Restrictions for Crypto Conditional Debug Support

-
- 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.

Note The **debug crypto condition peer** command with the **ipv4** or **ipv6** keyword can provide the hardware platform specific debugging output. The rest of the condition filters do not provide platform specific debugging output.

Table 7: 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.
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).
ikev2	The name string for an IKEv2 profile. Relevant debug messages will be shown if the IKEv2 profile name is specified.
isakmp	The name string for an ISAKMP profile. Relevant debug messages will be shown if the ISAKMP profile name is specified.
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).
local	The name string of an IPv4 or IPv6 local address.
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.

Condition Type (Keyword)	Description
peer ipv4 or peer ipv6	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.
session	Provides information about crypto sessions.
SPI	A 32-bit unsigned integer. Relevant debug messages will be shown if the current IPSec operation uses this value as the SPI.
unmatched	Provides debug messages when context information is unavailable.

[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.

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.
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:**	Enables global IKE debugging.

	Command or Action	Purpose
	`Router#` `debug crypto isakmp`	

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.

	Command or Action	Purpose
Step 2	**debug crypto isakmp | ipsec | engine} error** **Example:** `Router# debug crypto ipsec error`	Enables only error debugging messages for a crypto area.

Configuration Examples for the Crypto Conditional Debug CLIs

Enabling Crypto Conditional Debugging Example

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

```
Router#
debug crypto condition connid 2000 engine-id 1
Router#
debug crypto condition peer ipv4 10.1.1.1
Router#
debug crypto condition peer ipv4 10.1.1.2
Router#
debug crypto condition peer ipv4 10.1.1.3
Router#
debug crypto condition unmatched
! Verify crypto conditional settings.
Router#
show crypto debug-condition
Crypto conditional debug currently is turned ON
IKE debug context unmatched flag:ON
IPsec debug context unmatched flag:ON
Crypto Engine debug context unmatched flag:ON
IKE peer IP address filters:
10.1.1.1  10.1.1.2   10.1.1.3
Connection-id filters:[connid:engine_id]2000:1,
! Enable global crypto CLIs to start conditional debugging.
Router#
debug crypto isakmp
Router#
debug crypto ipsec
Router#
debug crypto engine
```

Disabling Crypto Conditional Debugging Example

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

```
Router#
debug crypto condition reset
! Verify that all crypto conditional settings have been disabled.
Router#
```

```
show crypto debug-condition
Crypto conditional debug currently is turned OFF
IKE debug context unmatched flag:OFF
IPsec debug context unmatched flag:OFF
Crypto Engine debug context unmatched flag:OFF
```

Additional References

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

Related Documents

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

Standards

Standards	Title
None	--

MIBs

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

RFCs

RFCs	Title
None	--

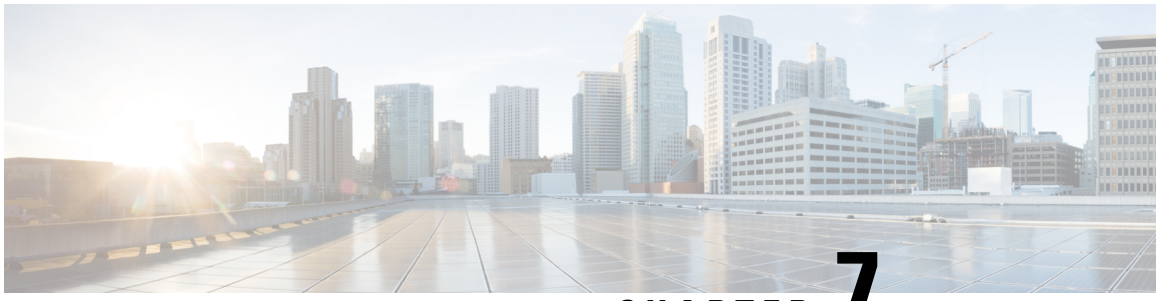
Technical Assistance

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

Feature Information for Crypto Conditional Debug Support

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

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



CHAPTER 7

IPv6 over IPv4 GRE Tunnel Protection

The IPv6 over IPv4 GRE Tunnel Protection feature allows both IPv6 unicast and multicast traffic to pass through a protected generic routing encapsulation (GRE) tunnel.

Finding Feature Information

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

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

Prerequisites for IPv6 over IPv4 GRE Tunnel Protection

- To enable this feature, you must configure IPsec tunnel protection on an IPv4 GRE tunnel.
- To enable IPv6 multicast, you must configure IPv6 multicast routing.

Restrictions for IPv6 over IPv4 GRE Tunnel Protection

The IPv6 over IPv4 GRE Tunnel Protection feature supports IPv6 over IPv4 point-to-point GRE tunnel protection and not IPv6 over IPv4 mGRE tunnel protection.

Information About IPv6 over IPv4 GRE Tunnel Protection

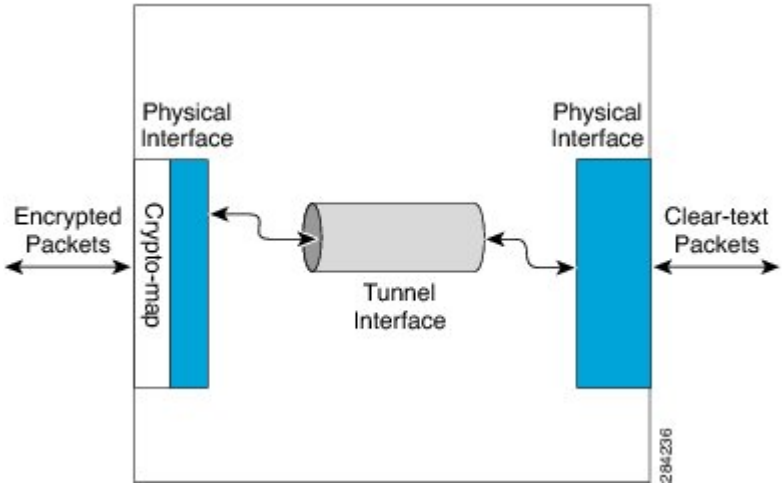
GRE Tunnels with IPsec

Generic routing encapsulation (GRE) tunnels sometimes are combined with IPSec, because IPSec does not support IPv6 multicast packets. This function prevents dynamic routing protocols from running successfully over an IPSec VPN network. Because GRE tunnels do support IPv6 multicast , a dynamic routing protocol can be run over a GRE tunnel. Once a dynamic routing protocol is configured over a GRE tunnel, you can encrypt the GRE IPv6 multicast packets using IPSec.

IPSec can encrypt GRE packets using a crypto map or tunnel protection. Both methods specify that IPSec encryption is performed after GRE encapsulation is configured. When a crypto map is used, encryption is applied to the outbound physical interfaces for the GRE tunnel packets. When tunnel protection is used, encryption is configured on the GRE tunnel interface.

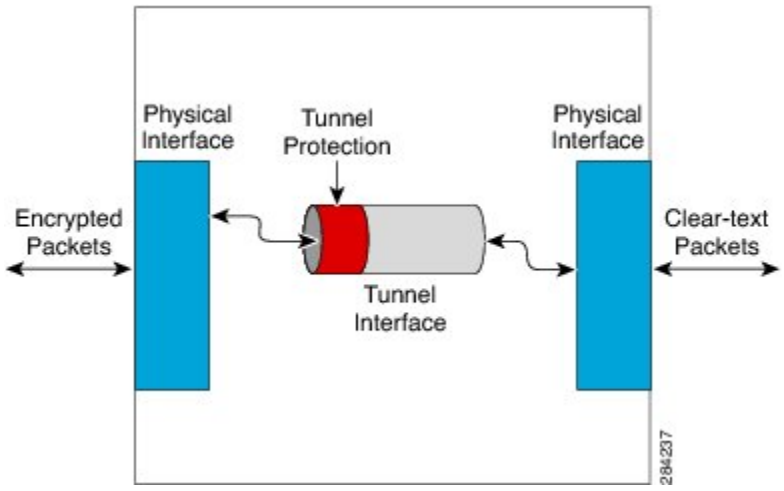
The following figure shows encrypted packets that enter a router through a GRE tunnel interface using a crypto map on the physical interface. Once the packets are decrypted and decapsulated, they continue to their IP destination as clear text.

Figure 8: Using a Crypto Map to Configure IPv6 over IPv4 GRE Tunnel Encryption



The following figure shows encryption using tunnel protection command on the GRE tunnel interface. The encrypted packets enter the router through the tunnel interface and are decrypted and decapsulated before they continue to their destination as clear text.

Figure 9: Using Tunnel Protection to Configure IPv6 over IPv4 GRE Tunnel Encryption



There are two key differences in using the crypto map and tunnel protection methods:

- The IPSec crypto map is tied to the physical interface and is checked as packets are forwarded out through the physical interface. At this point, the GRE tunnel has already encapsulated the packet.
- Tunnel protection ties the encryption functionality to the GRE tunnel and is checked after the packet is GRE encapsulated but before the packet is handed to the physical interface.

How to Configure IPv6 over IPv4 GRE Tunnel Protection

Configuring IPv6 over IPv4 GRE Encryption Using a Crypto Map

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 multicast-routing**
4. **ipv6 unicast-routing**
5. **interface** *type number*
6. **ipv6 address** {**ipv6-address**/*prefix-length* | **prefix-name** *sub-bits*/*prefix-length*}
7. **tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ip** | **gre ipv6** | **ipip** [**decapsulate-any**] | **ipsec ipv4** | **iptalk** | **ipv6** | **ipsec ipv6** | **mpls** | **nos** | **rbscp**}
8. **tunnel source** {**ip-address** | **ipv6-address** | *interface-typeinterface-number*}
9. **tunnel destination** {*hostname* | *ip-address* | *ipv6-address*}
10. **exit**
11. **crypto isakmp policy** *priority*
12. **authentication** {**rsa-sig** | **rsa-encr** | **pre-share**}
13. **hash** {**sha** | **md5**}
14. **group** {**1** | **2** | **5**}

15. **encryption** {**des** | **3des** | **aes 192** | **aes 256**}
16. **exit**
17. **crypto isakmp key** *enc-type-digit keystring* {**address** *peer-address* [*mask*] | **ipv6** {*ipv6-address/ipv6-prefix*} | **hostname** *hostname*} [**no-xauth**]
18. **crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*] [*transform4*]
19. **access-list** *access-list-number* [**dynamic** *dynamic-name* [**timeout** *minutes*]] {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**] [**log** [*word*] | **log-input** [*word*]]
20. **crypto map** [**ipv6**] *map-name seq-num* [**ipsec-isakmp** [**dynamic** *dynamic-map-name* | **discover** | **profile** *profile-name*]]
21. **set peer** {*hostname* [**dynamic**] [**default**] | *ip-address* [**default**]}
22. **set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*]
23. **match address** [*access-list-id* | *name*]
24. **exit**
25. **interface** *type number*
26. **crypto map** *map-name* [**redundancy** *standby-group-name* [**stateful**]]
27. **end**

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	**ipv6 multicast-routing** **Example:** `Router(config)# ipv6 multicast-routing`	Enables multicast routing using Protocol Independent Multicast (PIM) and Multicast Listener Discovery (MLD) on all IPv6-enabled interfaces of the router and enables multicast forwarding. • Enable this command only if you are using IPv6 multicast. If you are using IPv6 unicast, you need not enable this command.
Step 4	**ipv6 unicast-routing** **Example:** `Router(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 5	**interface** *type number* **Example:** `Router(config)# interface tunnel 10`	Specifies a tunnel interface and number, and enters interface configuration mode.

	Command or Action	Purpose
Step 6	**ipv6 address** {**ipv6-address**/*prefix-length* | **prefix-name** *sub-bits*/*prefix-length*} **Example:** `Router(config-if)# ipv6 address 0:0:0:7272::72/64`	Configures an IPv6 address based on an IPv6 general prefix and enables IPv6 processing on an interface.
Step 7	**tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ip** | **gre ipv6** | **ipip** [**decapsulate-any**] | **ipsec ipv4** | **iptalk** | **ipv6** | **ipsec ipv6** | **mpls** | **nos** | **rbscp**} **Example:** `Router(config-if)# tunnel mode gre ip`	Sets the encapsulation mode for the tunnel interface.
Step 8	**tunnel source** {**ip-address** | **ipv6-address** | *interface-typeinterface-number*} **Example:** `Router(config-if)# tunnel source ethernet0`	Sets the source address for a tunnel interface.
Step 9	**tunnel destination** {*hostname* | *ip-address* | *ipv6-address*} **Example:** `Router(config-if)# tunnel destination 172.16.0.12`	Specifies the destination for a tunnel interface.
Step 10	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns to global configuration mode.
Step 11	**crypto isakmp policy** *priority* **Example:** `Router(config)# crypto isakmp policy 15`	Defines an Internet Key Exchange (IKE) policy, and enters ISAKMP policy configuration mode. • Policy number 1 indicates the policy with the highest priority. The lower the *priority* argument value, the higher the priority.
Step 12	**authentication** {**rsa-sig** | **rsa-encr** | **pre-share**} **Example:** `Router(config-isakmp-policy)# authentication pre-share`	Specifies the authentication method within an IKE policy. • The **rsa-sig** and **rsa-encr** keywords are not supported in IPv6.
Step 13	**hash** {**sha** | **md5**} **Example:** `Router(config-isakmp-policy)# hash md5`	Specifies the hash algorithm within an IKE policy.
Step 14	**group** {**1** | **2** | **5**} **Example:** `Router(config-isakmp-policy)# group 2`	Specifies the Diffie-Hellman group identifier within an IKE policy.
Step 15	**encryption** {**des** | **3des** | **aes 192** | **aes 256**} **Example:**	Specifies the encryption algorithm within an IKE policy.

	Command or Action	Purpose
	`Router(config-isakmp-policy)# encryption 3des`	
Step 16	**exit** **Example:** `Router(config-isakmp-policy)# exit`	Exits ISAKMP policy configuration mode and enters global configuration mode.
Step 17	**crypto isakmp key** *enc-type-digit keystring* {**address** *peer-address* [*mask*] | **ipv6** {*ipv6-address*/*ipv6-prefix*} | **hostname** *hostname*} [**no-xauth**] **Example:** `Router(config)# crypto isakmp key cisco-10 address 172.16.0.12 255.240.0.0`	Configures a preshared authentication key.
Step 18	**crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*] [*transform4*] **Example:** `Router(config)# crypto ipsec transform-set myset0 ah-sha-hmac esp-3des`	Defines a transform set.
Step 19	**access-list** *access-list-number* [**dynamic** *dynamic-name* [**timeout** *minutes*]] {**deny** | **permit**} *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*] [**tos** *tos*] [**time-range** *time-range-name*] [**fragments**] [**log** [*word*] | **log-input** [*word*]] **Example:** `Router(config)# access-list 110 permit gre host 192.168.0.16 host 172.16.0.12`	Defines an extended IP access list.
Step 20	**crypto map** [**ipv6**] *map-name seq-num* [**ipsec-isakmp** [**dynamic** *dynamic-map-name* | **discover** | **profile** *profile-name*]] **Example:** `Router(config)# crypto map mymap 10 ipsec-isakmp`	Creates a new crypto map entry or profile and enters crypto map configuration mode.
Step 21	**set peer** {*hostname* [**dynamic**] [**default**] | *ip-address* [**default**]} **Example:** `Router(config-crypto-map)# set peer 10.0.0.1`	Specifies an IP Security (IPsec) peer in a crypto map entry.
Step 22	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Router(config-crypto-map)# set transform-set myset0`	Specifies the transform set that can be used with the crypto map entry.

	Command or Action	Purpose
Step 23	**match address** [*access-list-id* | *name*] **Example:** `Router(config-crypto-map)# match address 102`	Specifies an extended access list for a crypto map entry.
Step 24	**exit** **Example:** `Router(config-crypto-map)# exit`	Exits crypto map configuration mode and returns to global configuration mode.
Step 25	**interface** *type number* **Example:** `Router(config)# interface ethernet 1`	Specifies an interface and number and enters interface configuration mode.
Step 26	**crypto map** *map-name* [**redundancy** *standby-group-name* [**stateful**]] **Example:** `Router(config-if)# crypto map mymap`	Applies a previously defined crypto map set to an outbound interface.
Step 27	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuring IPv6 over IPv4 GRE Encryption Using Tunnel Protection

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ipv6 multicast-routing**
4. **ipv6 unicast-routing**
5. **crypto isakmp policy** *priority*
6. **authentication** {**rsa-sig** | **rsa-encr** | **pre-share**}
7. **hash** {**sha** | **md5**}
8. **group** {**1** | **2** | **5**}
9. **encryption** {**des** | **3des** | **aes** | **aes 192** | **aes 256**}
10. **exit**
11. **crypto isakmp key** *enc-type-digit keystring* {**address** *peer-address* [*mask*] | **ipv6** {*ipv6-address*/*ipv6-prefix*} | **hostname** *hostname*} [**no-xauth**]
12. **crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*] [*transform4*]
13. **crypto ipsec profile** *profile-name*
14. **set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*]
15. **exit**
16. **interface** *type number*
17. **ipv6 address** {*ipv6-address* / *prefix-length* | *prefix-name sub-bits*/*prefix-length*}

18. **tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ip** | **gre ipv6** | **ipip**[**decapsulate-any**] | **ipsec ipv4** | **iptalk** | **ipv6** | **ipsec ipv6** | **mpls** | **nos** | **rbscp**}
19. **tunnel source** {*ip-address* | *ipv6-address* | *interface-type interface-number*}
20. **tunnel destination** {*hostname* | *ip-address* | *ipv6-address*}
21. **tunnel protection ipsec profile** *name* [**shared**]
22. **end**

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	**ipv6 multicast-routing** **Example:** `Router(config)# ipv6 multicast-routing`	Enables multicast routing using Protocol Independent Multicast (PIM) and Multicast Listener Discovery (MLD) on all IPv6-enabled interfaces of the router and enables multicast forwarding. • Enable this command only if you are using IPv6 multicast. If you are using IPv6 unicast, you do not need to enable this command.
Step 4	**ipv6 unicast-routing** **Example:** `Router(config)# ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams.
Step 5	**crypto isakmp policy** *priority* **Example:** `Router(config)# crypto isakmp policy 15`	Defines an IKE policy, and enters ISAKMP policy configuration mode. Policy number 1 indicates the policy with the highest priority. The lower the *priority* argument value, the higher the priority.
Step 6	**authentication** {**rsa-sig**	**rsa-encr**	**pre-share**} **Example:** `Router(config-isakmp-policy)# authentication pre-share`	Specifies the authentication method within an Internet Key Exchange (IKE) policy. • The **rsa-sig** and **rsa-encr** keywords are not supported in IPv6.
Step 7	**hash** {**sha**	**md5**} **Example:** `Router(config-isakmp-policy)# hash md5`	Specifies the hash algorithm within an IKE policy.

	Command or Action	Purpose
Step 8	**group** {**1** | **2** | **5**} **Example:** `Router(config-isakmp-policy)# group 2`	Specifies the Diffie-Hellman group identifier within an IKE policy.
Step 9	**encryption** {**des** | **3des** | **aes** | **aes 192** | **aes 256**} **Example:** `Router(config-isakmp-policy)# encryption 3des`	Specifies the encryption algorithm within an IKE policy.
Step 10	**exit** **Example:** `Router(config-isakmp-policy)# exit`	Exits ISAKMP policy configuration mode and returns to global configuration mode.
Step 11	**crypto isakmp key** *enc-type-digit keystring* {**address** *peer-address* [*mask*] | **ipv6** {*ipv6-address/ipv6-prefix*} | **hostname** *hostname*} [**no-xauth**] **Example:** `Router(config)# crypto isakmp key cisco-10 address 172.16.0.12 255.240.0.0`	Configures a preshared authentication key.
Step 12	**crypto ipsec transform-set** *transform-set-name transform1* [*transform2*] [*transform3*] [*transform4*] **Example:** `Router(config)# crypto ipsec transform-set myset0 ah-sha-hmac esp-3des`	Defines a transform set, and places the router in crypto transform configuration mode.
Step 13	**crypto ipsec profile** *profile-name* **Example:** `Router(config)# crypto ipsec profile ipsecprof`	Defines the IPsec parameters that are to be used for IPsec encryption between two IPsec routers and enters IPsec profile configuration mode.
Step 14	**set transform-set** *transform-set-name* [*transform-set-name2...transform-set-name6*] **Example:** `Router(ipsec-profile)# set transform-set myset0`	Specifies the transform set that can be used with the crypto map entry.
Step 15	**exit** **Example:** `Router(ipsec-profile)# exit`	Exits IPsec profile configuration mode and returns to global configuration mode.
Step 16	**interface** *type number* **Example:** `Router(config)# interface tunnel 1`	Specifies a tunnel interface and number and enters interface configuration mode.
Step 17	**ipv6 address** {*ipv6-address* / *prefix-length* | *prefix-name sub-bits/prefix-length*} **Example:**	Specifies the IPv6 network assigned to the interface and enables IPv6 processing on the interface.

	Command or Action	Purpose
	`Router(config-if)# ipv6 address 3ffe:b00:c18:1::3/127`	
Step 18	**tunnel mode** {**aurp** | **cayman** | **dvmrp** | **eon** | **gre** | **gre multipoint** | **gre ip** | **gre ipv6** | **ipip**[**decapsulate-any**] | **ipsec ipv4** | **iptalk** | **ipv6** | **ipsec ipv6** | **mpls** | **nos** | **rbscp**} **Example:** `Router(config-if)# tunnel mode gre ip`	Specifies a GRE IPv6 tunnel.
Step 19	**tunnel source** {*ip-address* | *ipv6-address* | *interface-type interface-number*} **Example:** `Router(config-if)# tunnel source 10.0.0.1`	Specifies the source address or the source interface type and number for the tunnel interface.
Step 20	**tunnel destination** {*hostname* | *ip-address* | *ipv6-address*} **Example:** `Router(config-if)# tunnel destination 172.16.0.12`	Specifies the destination address or hostname for the tunnel interface.
Step 21	**tunnel protection ipsec profile** *name* [**shared**] **Example:** `Router(config-if)# tunnel protection ipsec profile ipsecprof`	Associates a tunnel interface with an IPsec profile.
Step 22	**end** **Example:** `Router(config-if)# end`	Exits interface configuration mode and returns to privileged EXEC mode.

Configuration Examples for IPv6 over IPv4 GRE Tunnel Protection

Example: Configuring IPv6 over IPv4 GRE Encryption Using a Crypto Map

```
Router> enable
Router# configure terminal
Router(config)# ipv6 multicast-routing
Router(config)# ipv6 unicast-routing
Router(config)# interface tunnel 10
Router(config-if)# ipv6 address my-prefix 0:0:0:7272::72/64
Router(config-if)# tunnel mode gre ip
Router(config-if)# tunnel source ethernet0
Router(config-if)# tunnel destination 172.16.0.12
Router(config-if)# exit
Router(config)# crypto isakmp policy 15
Router(config-isakmp-policy)# authentication pre-share
Router(config-isakmp-policy)# hash md5
Router(config-isakmp-policy)# group 2
Router(config-isakmp-policy)# encryption 3des
Router(config-isakmp-policy)# exit
Router(config)# crypto isakmp key cisco-10 address 172.16.0.12 255.240.0.0
```

```
Router(config)# crypto ipsec transform-set myset0 ah-sha-hmac esp-3des
Router(config)# access-list 110 permit gre host 192.168.0.16 host 172.16.0.12
Router(config)# crypto map mymap 10 ipsec-isakmp
Router(config-crypto-map)# set peer 10.0.0.1
Router(config-crypto-map)# set transform-set myset0
Router(config-crypto-map)# match address 102
Router(config-crypto-map)# exit
Router(config)# interface ethernet1
Router(config-if)# crypto map mymap
Router(config-if)# end
```

Example: Configuring IPv6 over IPv4 GRE Encryption Using Tunnel Protection

The following example configures IPsec tunnel protection on an IPv4 GRE tunnel. IPv6 multicast routing is enabled using the **ipv6 multicast-routing** command.

```
Router> enable
Router# configure terminal
Router(config)# ipv6 multicast-routing
Router(config)# ipv6 unicast-routing
Router(config)# crypto isakmp policy 15
Router(config-isakmp-policy)# authentication pre-share
Router(config-isakmp-policy)# hash md5
Router(config-isakmp-policy)# group 2
Router(config-isakmp-policy)# encryption 3des
Router(config-isakmp-policy)# exit
Router(config)# crypto isakmp key cisco-10 address 172.16.0.12 255.240.0.0
Router(config)# crypto ipsec transform-set myset0 ah-sha-hmac esp-3des
Router(config)# crypto ipsec profile ipsecprof
Router(ipsec-profile)# set transform-set myset0
Router(ipsec-profile)# exit
Router(config)# interface tunnel 1
Router(config-if)# ipv6 address 3ffe:b00:c18:1::3/127
Router(config-if)# tunnel mode gre ip
Router(config-if)# tunnel source 10.0.0.1
Router(config-if)# tunnel destination 172.16.0.12
Router(config-if)# tunnel protection ipsec profile ipsecprof
Router(config-if)# end
```

Additional References

Related Documents

Related Topic	Document Title
IPv6 Multicast Routing	IPv6 Implementation Guide
Cisco IOS commands	*Cisco IOS Master Commands List, All Releases*

Related Topic	Document Title
Security commands: complete command syntax, command mode, command history, 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*

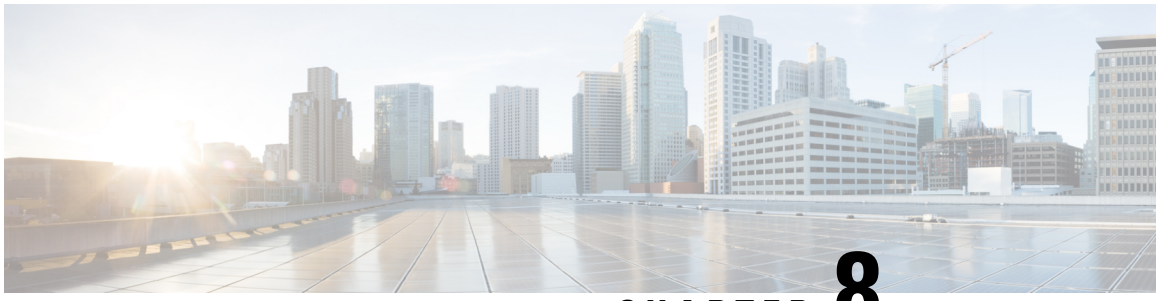
Technical Assistance

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 IPv6 over IPv4 GRE Tunnel Protection

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.



CHAPTER 8

RFC 430x IPsec Support

The RFC 430x IPsec Support includes features—RFC 430x IPsec Support Phase 1 and RFC430x IPsec Support Phase 2—that implement Internet Key Exchange (IKE) and IPsec behavior as specified in RFC 4301.

Finding Feature Information

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

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

Information About RFC 430x IPsec Support

RFC 430x IPsec Support Phase 1

The RFC 430x IPsec Support Phase 1 feature implements Internet Key Exchange (IKE) and IPsec behavior as specified in RFC 4301.

RFC 4301 specifies the base architecture for IPsec-compliant systems. RFC 4301 describes how to provide a set of security services for traffic at the IP layer, in both the IPv4 and IPv6 environments. The RFC 430x IPsec Support Phase 1 feature provides support for the following RFC 4301 implementations on Cisco IOS software.

- **Security association (SA) lifetime**—The lifetime of a security association between IPsec and Internet Key Exchange (IKE) or Internet Key Exchange Version 2 (IKEv2) must not exceed the lifetime of the authentication certificate.

- **OPAQUE selectors**—OPAQUE indicates that the corresponding selector field is not available for verification. When IKEv2 encounters an OPAQUE selector, IKEv2 skips, does not process the OPAQUE selector, and moves to next selector for policy verification.
- **Explicit Congestion Notification (ECN) support**—ECN is propagated when decrypting an IPsec packet thereby ensuring the packet source and destination are aware of congestion that occurs within the network.
- **Fragment processing**—Peers must not send Initial and noninitial fragments in the same tunnel. There must be a separate tunnel mode SA for carrying initial and noninitial fragments and separate tunnel mode SA for noninitial fragments. IPsec peers must support discarding of packets and stateful fragment checking to accommodate bypass traffic.
- **Do not fragment-(DF) bit processing**—DF-bit processing must be set on a per SA basis.
- **Dummy packet generation support**—It should be possible to send dummy packets via IPsec SA to encapsulate the packets when traffic is flowing via IPsec SA tunnel.

RFC 430x IPsec Support Phase 2

The RFC 430x IPsec Support Phase 2 feature provides support for the RFC 4301 implementation of encryption and decryption of Internet Control Message Protocol (ICMP) packets on Cisco IOS software.

ICMP error messages are sent when an ICMP error occurs. For example, when a host is not reachable, the intermediate device sends a message to the originator of the ICMP request that the host is not reachable. When an ICMP error message reaches an IPsec encryption policy, it may not be classified to match an existing SA. So, the packets are classified based on the data inside the ICMP error message. This data contains the source and destination address of the original ICMP message. If an SA is found based on the address in the ICMP error message, the SA is used. If there is no SA, an SA is created if the policy permits. For decryption, the post decrypt check is performed on the data inside the ICMP error message if a valid SA is not found.

The encryption and decryption of ICMP error messages can be verified through the encrypt and decrypt counters displayed in the output of the **show crypto ipsec sa** command.

How to Configure RFC 430x IPsec Support

Configuring RFC 430x IPsec Support Globally

Perform this task to configure the RFC 4301 implementations globally.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto ipsec security-association dummy** {**pps** *rate* | **seconds** *seconds*}
4. **crypto ipsec security-association ecn** {**discard** | **propogate**}
5. **exit**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**crypto ipsec security-association dummy** {**pps** *rate* | **seconds** *seconds*} **Example:** `Device(config)# crypto ipsec security-association dummy seconds 5`	Enables the generation and transmission of dummy packets in an IPsec traffic flow.
Step 4	**crypto ipsec security-association ecn** {**discard** | **propogate**} **Example:** `Device(config)# crypto ipsec security-association ecn discard`	Enables the Explicit Congestion Notification (ECN) settings in an IPsec traffic flow.
Step 5	**exit** **Example:** `Device(config-crypto-map)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Configuring RFC 430x IPsec Support Per Crypto Map

Perform this task to configure the RFC 4301 implementations per crypto map.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto map** *map-name seq-num* **ipsec-isakmp**
4. **set ipsec security-association dfbit** {**clear** | **copy** | **set**}
5. **set ipsec security-association dummy** {**pps** *rate* | **seconds** *seconds*}
6. **set ipsec security-association ecn** {**discard** | **propogate**}
7. **end**
8. **show crypto map ipsec sa**

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 map** *map-name seq-num* **ipsec-isakmp** **Example:** `Device(config)# crypto map cmap 1 ipsec-isakmp`	Specifies the crypto map entry to be created or modified and enters crypto map configuration mode.
Step 4	**set ipsec security-association dfbit** {**clear** | **copy** | **set**} **Example:** `Device(config-crypto-map)# set ipsec security-association dfbit set`	Enables do not fragment (DF)-bit processing per security association (SA) for an IPsec traffic flow in a crypto map.
Step 5	**set ipsec security-association dummy** {**pps** *rate* | **seconds** *seconds*} **Example:** `Device(config-crypto-map)# set ipsec security-association dummy seconds 5`	Enables the generation and transmission of dummy packets for an IPsec traffic flow in a crypto map.
Step 6	**set ipsec security-association ecn** {**discard** | **propogate**} **Example:** `Device(config-crypto-map)# set ipsec security-association ecn propogate`	Enables the Explicit Congestion Notification (ECN) settings per SA for an IPsec traffic flow in a crypto map.
Step 7	**end** **Example:** `Device(config-crypto-map)# end`	Exits crypto map configuration mode and returns to privileged EXEC mode.
Step 8	**show crypto map ipsec sa** **Example:** `Device# show crypto map ipsec sa`	Displays the settings used by IPsec SAs.

Example

The following is sample output from the **show crypto map ipsec sa** command:

```
Device# show crypto map ipsec sa

interface: Tunnel0
 Crypto map tag: Tunnel0-head-0, local addr 3FFE:2002::32F7:DFF:FE54:7FD1
protected vrf: (none)
local ident (addr/mask/prot/port): (3FFE:2002::32F7:DFF:FE54:7FD1/128/47/0)
remote ident (addr/mask/prot/port): (3FFE:2002::C671:FEFF:FE88:EB82/128/47/0)
current_peer 3FFE:2002::C671:FEFF:FE88:EB82 port 500
 PERMIT, flags={origin_is_acl,}
#pkts encaps: 36, #pkts encrypt: 36, #pkts digest: 36
#pkts decaps: 28, #pkts decrypt: 28, #pkts verify: 28
#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
```

```
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0
#send dummy packets 852600, #recv dummy packets 424905

local crypto endpt.: 3FFE:2002::32F7:DFF:FE54:7FD1,
remote crypto endpt.: 3FFE:2002::C671:FEFF:FE88:EB82
plaintext mtu 1430, path mtu 1500, ipv6 mtu 1500, ipv6 mtu idb GigabitEthernet0/0/1
current outbound spi: 0xE963D1EC(3915633132)
PFS (Y/N): N, DH group: none
Dummy packet: Initializing

inbound esp sas:
spi: 0xF4E01B9A(4108327834)
 transform: esp-3des esp-md5-hmac,
 in use settings ={Tunnel, }
 conn id: 2053, flow_id: ESG:53, sibling_flags FFFFFFFF80000049, crypto map: Tunnel0-head-0

 sa timing: remaining key lifetime (k/sec): (4608000/2343)
 IV size: 8 bytes
 replay detection support: Y
 Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:

outbound esp sas:
spi: 0xE963D1EC(3915633132)
 transform: esp-3des esp-md5-hmac,
 in use settings ={Tunnel, }
 conn id: 2054, flow_id: ESG:54, sibling_flags FFFFFFFF80000049, crypto map: Tunnel0-head-0

 sa timing: remaining key lifetime (k/sec): (4608000/2343)
 IV size: 8 bytes
 replay detection support: Y
 Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:
```

Configuration Examples for RFC 430x IPsec Support

Example: Configuring RFC 430x IPsec Support Globally

The following examples shows how to configure RFC 430x IPsec Support globally:

```
Device> enable
Device# configure terminal
Device(config)# crypto ipsec security-association dummy seconds 15
Device(config)# crypto ipsec security-association ecn propogate
Device(config-crypto-map)# exit
```

Example: Configuring RFC 430x IPsec Support Per Crypto Map

The following examples shows how to configure RFC 430x IPsec Support per crypto map:

```
Device> enable
Device# configure terminal
Device(config)# crypto map cmap 1 ipsec-isakmp
Device(config-crypto-map)# set security-association copy
Device(config-crypto-map)# set security-association dummy seconds 15
Device(config-crypto-map)# set security-association ecn propogate
Device(config-crypto-map)# end
Device# show crypto map ipsec sa

interface: Tunnel0
 Crypto map tag: Tunnel0-head-0, local addr 3FFE:2002::32F7:DFF:FE54:7FD1
protected vrf: (none)
local ident (addr/mask/prot/port): (3FFE:2002::32F7:DFF:FE54:7FD1/128/47/0)
remote ident (addr/mask/prot/port): (3FFE:2002::C671:FEFF:FE88:EB82/128/47/0)
current_peer 3FFE:2002::C671:FEFF:FE88:EB82 port 500
 PERMIT, flags={origin_is_acl,}
#pkts encaps: 36, #pkts encrypt: 36, #pkts digest: 36
#pkts decaps: 28, #pkts decrypt: 28, #pkts verify: 28
#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0
#send dummy packets 852600, #recv dummy packets 424905

local crypto endpt.: 3FFE:2002::32F7:DFF:FE54:7FD1,
remote crypto endpt.: 3FFE:2002::C671:FEFF:FE88:EB82
plaintext mtu 1430, path mtu 1500, ipv6 mtu 1500, ipv6 mtu idb GigabitEthernet0/0/1
current outbound spi: 0xE963D1EC(3915633132)
PFS (Y/N): N, DH group: none
Dummy packet: Initializing

inbound esp sas:
spi: 0xF4E01B9A(4108327834)
 transform: esp-3des esp-md5-hmac,
 in use settings ={Tunnel, }
 conn id: 2053, flow_id: ESG:53, sibling_flags FFFFFFFF80000049, crypto map: Tunnel0-head-0

 sa timing: remaining key lifetime (k/sec): (4608000/2343)
 IV size: 8 bytes
 replay detection support: Y
 Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:

outbound esp sas:
spi: 0xE963D1EC(3915633132)
 transform: esp-3des esp-md5-hmac,
 in use settings ={Tunnel, }
 conn id: 2054, flow_id: ESG:54, sibling_flags FFFFFFFF80000049, crypto map: Tunnel0-head-0

 sa timing: remaining key lifetime (k/sec): (4608000/2343)
 IV size: 8 bytes
 replay detection support: Y
 Status: ACTIVE(ACTIVE)

outbound ah sas:
```

```
outbound pcp sas:
```

Additional References for RFC 430x IPsec Support

Related Documents

Related Topic	Document Title
Cisco IOS Commands	Cisco IOS Master Command 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
IKEv2 configuration	
Recommended cryptographic algorithms	Next Generation Encryption

Standards and RFCs

Standard/RFC	Title
RFC 4301	*Security Architecture for the Internet Protocol*

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 RFC 430x IPsec Support

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

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

Table 8: Feature Information for RFC430x IPsec Support

Feature Name	Releases	Feature Information
RFC430x IPsec Support Phase 1		The RFC 430x IPsec Support Phase 1 feature implements Internet Key Exchange (IKE) and IPsec behavior as specified in RFC 4301. The following commands were introduced or modified: **crypto ipsec security-association dummy, crypto ipsec security-association ecn, set ipsec security-association dfbit, set ipsec security-association dummy, set ipsec security-association ecn, show crypto map ipsec sa.**
RFC430x IPsec Support Phase 2		The RFC 430x IPsec Support Phase 2 feature provides support for the RFC 4301 implementation of encryption and decryption of Internet Control Message Protocol (ICMP) packets on Cisco IOS software. No commands were modified or updated for this feature.