

# Configuring IPSec

# 10

This chapter describes IPSec capabilities of the ERX system.

Topic	Page
Overview	10-1
References	10-3
IPSec Concepts	10-4
IKE Overview	10-16
Configuration Tasks	10-20
Configuration Examples	10-29
Monitoring IPSec	10-40

## Overview

---

The IP security functionality covered in this chapter includes the following major areas:

- Encapsulating protocols, including AH and ESP, to provide security on specified packets
- The ISAKMP/IKE protocol suite to provide automatic negotiation of security associations including session keys

*Terms*

Table 10-1 defines terms and abbreviations that are used in this discussion of IPSec.

**Table 10-1** IPSec terms and abbreviations

Term	Definition
3DES	Triple DES encryption/decryption algorithm
AH	Authentication Header. Provides authentication of the sender and of data integrity.
CA	certificate authority
DES	Data Encryption Standard encryption algorithm
DSS	Digital Signature Standard authentication algorithm
ESP	Encapsulating Security Payload, provides data integrity, data confidentiality and, optionally, sender's authentication
HMAC	Hashed Message Authentication Code
IKE	Internet Key Exchange
IKE endpoint	IP address of the entity that is one of two endpoints in an IKE/ISAKMP SA.
Inbound traffic	In the context of a secure interface, inbound traffic refers to already secured traffic arriving on that interface (identified based on its SPI). This traffic is cleared and checked against the security parameters set for that interface.
IPSec	IP security
IPSec endpoint	IP address of the entity that is one of two endpoints in an IPSec SA
ISAKMP	Internet Security Association and Key Management Protocol
ISAKMP SA	Security associations used to secure control channels between security gateways. These are negotiated via IKE phase I.
MDx	Message Digest x hash algorithm
Nonce	A random value used to detect and protect against replay attacks
Outbound traffic	In the context of a secure interface, outbound traffic refers to the clear traffic forwarded to that interface (either by policy or by routing) that should be secured according to security parameters set for that interface.
PFS	Perfect forward secrecy
RSA	Rivest-Shamir-Adleman encryption algorithm
SA	Security association. The set of security parameters that dictate how IPSec processes a packet, including encapsulation protocol and session keys. A single secure tunnel uses multiple SAs.

**Table 10-1** IPSec terms and abbreviations (continued)

Term	Definition
Secure tunnel	A virtual connection between two security gateways used to exchange data packets in a secure way. A secure tunnel is made up of a local SA and a remote SA, where both are negotiated in the context of an ISAKMP SA.
SHA	Secure Hash Algorithm
SPI	Security parameter index
VPN	Virtual private network

## References

---

This IPSec implementation supports the following RFCs:

- RFC 2401 – Security Architecture for the Internet Protocol (November 1998)
- RFC 2402 – IP Authentication Header (November 1998)
- RFC 2403 – The Use of HMAC-MD5-96 within ESP and AH (November 1998)
- RFC 2404 – The Use of HMAC-SHA-1-96 within ESP and AH (November 1998)
- RFC 2405 – The ESP DES-CBC Cipher Algorithm With Explicit IV (November 1998)
- RFC 2406 – IP Encapsulating Security Payload (ESP) (November 1998)
- RFC 2407 – The Internet IP Security Domain of Interpretation for ISAKMP (November 1998)
- RFC 2408 – Internet Security Association and Key Management Protocol (ISAKMP) (November 1998)
- RFC 2409 – The Internet Key Exchange (IKE) (November 1998 )
- RFC 2410 – The NULL Encryption Algorithm and Its Use With IPSec (November 1998)
- RFC 768 – User Datagram Protocol (August 1980)

## IPSec Concepts

---

This section provides an overview of IPSec concepts.

IPSec provides security to IP flows through the use of authentication and encryption.

- Authentication verifies that data is not altered during transmission and ensures that users are communicating with the individual or organization that they believe they are communicating with.
- Encryption makes data confidential by making it unreadable to everyone except the sender and intended recipient.

IPSec comprises two encapsulating protocols:

- Encapsulating Security Payload (ESP) provides confidentiality and authentication functions to every data packet.
- Authentication Header (AH) provides authentication to every data packet.

### *Secure IP Interfaces*

Secure IP interfaces are virtual IP interfaces that can be configured to provide confidentiality and authentication services for the data flowing through such interfaces. The software provides these services using mechanisms created by the suite of IPSec standards established by the IETF. However, the ERX system does not implement all mechanisms mandated by IPSec. Therefore, secure IP interfaces are not IPSec compliant.

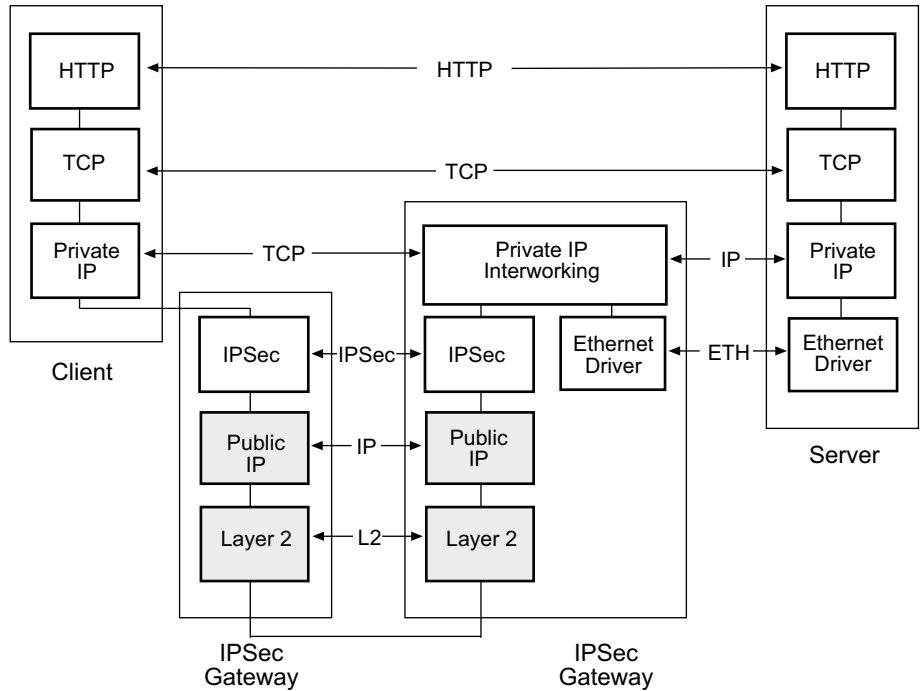
Secure IP interfaces connect the system to any other endpoint through the routed network and allow much of the same functionality as other IP interfaces. Traffic can reach a secure IP interface via routing and/or policy routing.

- A secure tunnel is a layer 2 entity. It is a point-to-point connection that is mapped on top of other IP interfaces. Secure tunnels carry only IP traffic.
- A secure IP interface is a layer 3 entity; that is, an IP interface mapped on top of a secure tunnel that inherits all security associated with it.

Secure IP interfaces are a logical representation of a secure connection between two security endpoints, one of which is the local system. The remote endpoint can be another security gateway or a host.

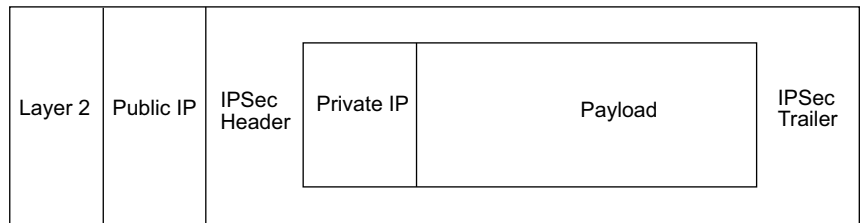
### IPSec Protocol Stack

Figure 10-1 shows the protocol stack on a client, an IPSec gateway, and a server. In the figure, HTTP and TCP are examples of higher-level protocols involved in the end-to-end communication; other end-to-end communication protocols are also supported. The layers where the data can be encrypted are shown in gray.



**Figure 10-1** IPSec tunneling stack

Figure 10-2 shows the packet encapsulation for IPSec tunneling.



**Figure 10-2** IPSec tunneling packet encapsulation

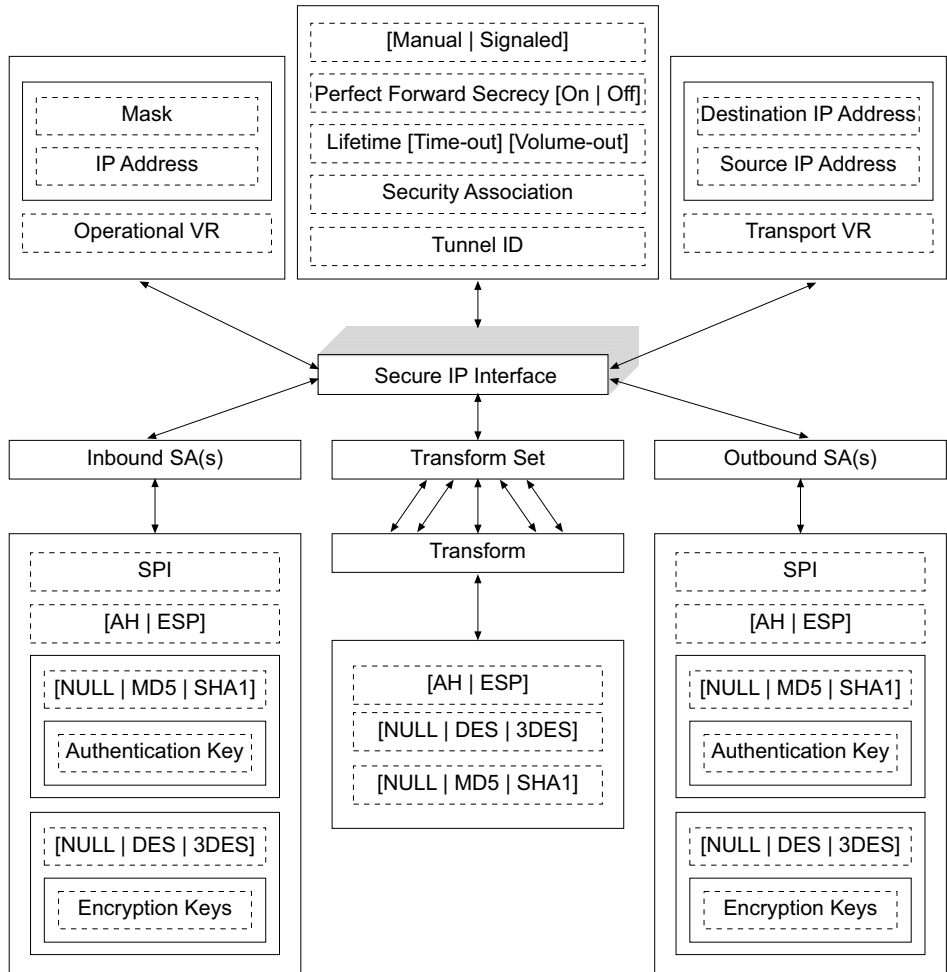
*Security Parameters*

Secure IP interfaces allow tunneled traffic to be secured in many ways. For that, secure interfaces are associated with security parameters that are enforced for traffic that goes through these interfaces. Table 10-2 gives a brief description of all parameters used for a secure IP interface.

**Table 10-2** Security parameters used on secure IP interfaces

<b>Security Parameter</b>	<b>Description</b>
Manual or signaled	A secure IP interface can be either manual or signaled. <ul style="list-style-type: none"><li>• Manual interfaces are configured manually on both local and remote security gateways.</li><li>• Signaled interfaces are able to dynamically set up connections between security gateways using ISAKMP/IKE.</li></ul>
Operational VR	Operational parameters for the secure IP interface, including the virtual router context to which this interface belongs and the network prefix reachable through the interface.
Transport VR	Transport network characteristics for the tunnel, including its virtual router context and source and destination IP addresses.
Perfect forward secrecy	A key generation approach that guarantees that every newly generated session key is not in any way related to the previous keys. PFS ensures that a compromised session key will not compromise previous and subsequent keys.
Lifetime	A limit on time and traffic volume allowed over the interface before an SA needs to be renegotiated.
Inbound and outbound SAs	The actual session-related parameters used by both security gateways to secure the traffic between them. The SA can be manually defined for manual secure IP tunnels or dynamically negotiated for signaled tunnels.  Two sets of SA parameters exist. One for inbound traffic and another for outbound traffic.
Transform set	The set of security parameters, including protocols and algorithms, that is considered adequate to provide a required security level to the traffic flowing through an interface.

Figure 10-3 shows the relationships of the various security parameters to the IPSec security interface. Each parameter is covered in detail in the following sections.



**Figure 10-3** IPSec security parameters in relation to secure IP interface

### Manual Versus Signaled Interfaces

The system supports both manual and signaled interfaces:

- Manual interfaces use a preconfigured set of SA parameters to secure traffic flowing through a secure IP interface. If SA parameters are not preconfigured for a manual secure interface, all traffic reaching the interface is dropped. Statistics are kept for dropped traffic. A manual secure IP tunnel must be manually provisioned in both peer security gateways.

- Signaled interfaces negotiate an SA on demand with the remote security gateway. The remote security gateway must also support SA negotiation; otherwise traffic is dropped. Again, statistics are kept for dropped traffic.

The system supports SA negotiation within an IKE SA by means of the ISAKMP and IKE protocols. Only one IKE SA is maintained between a set of local and remote IKE endpoints. That means that if an IKE SA already exists between the two endpoints, it is reused.

Secure IP interface parameters can be required, optional, or not applicable, depending on whether the interface is manual or signaled. Table 10-3 shows how the other security parameters fit with manual and signaled interfaces.

**Table 10-3** Security parameters per IPsec policy type

Security Parameter	Manual	Signaled
Transport VR	Required	Required
Operational VR	Required	Required
Perfect forward secrecy	Optional	Optional
Lifetime	Optional	Optional
Inbound and outbound SAs	Required	Not Applicable
Transform set	Required	Required

### Operational Virtual Router

The operational VR for a secure IP tunnel is the VR in which a secure IP tunnel exists.

The IP address and mask associated with a secure IP interface exist only within the operational VR under which the interface is declared. The VR defines the network prefix, which is reachable through the logical IP interface.

A secure IP tunnel is always a member of one and only one operational VR. Therefore, the operational VR attributes are mandatory for any secure tunnel. These attributes include:

- IP address and mask
- Virtual router where the secure IP interface exists

### Transport Virtual Router

The transport VR for a secure IP tunnel is the VR in which both of the secure tunnel endpoints, the source and destination, are routable addresses. Normally, the transport VR is the default ISP routing infrastructure on top of which VPNs are provisioned.

The IPSec service line module is a security gateway and, as such, is one of the endpoints for secure tunnels. The tunnel endpoints are the tunnel *source* and the tunnel *destination* IP addresses.

- The tunnel source IP address must be one of the local IP addresses configured on the system.
- The tunnel destination address must be a routable IP address within the transport VR routing tables.

The transport VR information is required, although its explicit configuration is not. If omitted, the transport VR is assumed to be the same as the operational VR. However, the tunnel source and destination are mandatory elements.

**Transport VR Definition** The transport VR definition includes:

- Transport virtual router name – If not explicitly configured, the operational VR is assumed.
- Tunnel source endpoint – IP address used as the tunnel source endpoint on this end of the tunnel. In the case of signalled tunnels, the system monitors and transmits on port 500 of this address for IKE negotiations. The tunnel source endpoint must be a configured IP address on the transport VR, or the system indicates an error.
- Tunnel destination endpoint – IP address associated with the termination or initiation point of the secure IP tunnel. This address must be routable within the context of the transport VR. Each secure IP tunnel can have a different remote IP address.

### Perfect Forward Secrecy

PFS is an optional feature that causes every newly refreshed key to be completely unrelated to the previous key. PFS provides added security, but requires extra processing for a new Diffie-Hellmann key exchange on every key refresh.

If PFS is enabled, the system mandates PFS during SA negotiation. The remote security gateway must accept PFS if the SA is to be successfully negotiated.

On the other hand, if PFS is disabled, PFS may still be negotiated if the remote security gateway requests PFS.

PFS supports three Diffie-Hellmann prime modulus groups:

- Group 1 – A 768-bit Diffie-Hellmann prime modulus group
- Group 2 – A 1024-bit Diffie-Hellmann prime modulus group
- Group 5 – A 1536-bit Diffie-Hellmann prime modulus group

SA negotiation favors the highest request. For example, if group 2 is requested locally, the remote security gateway must support group 2 for the SA negotiation to be successful. If group 1 is requested locally, then either groups 1 or 2 can be accepted, depending on requests from the remote security gateway.

### Lifetime

You can set a lifetime for user SAs and IKE SAs. For information on setting the IKE SA lifetime, see *Lifetime* in the *IKE Policies* section later in this chapter.

For signaled IPSec interfaces, both the inbound and outbound SA must be assigned a lifetime. The lifetime parameter controls the duration for which the SA is valid. When a user SA is established, both a timer and a traffic volume counter are set. When either counter reaches the limit specified by the SA's lifetime, a new SA is negotiated and the expired SA is deleted. The renegotiations refresh several SA parameters, including keys.

Note the following about how the lifetime parameters work:

- To avoid delays in the data flow, a new user SA is actually renegotiated before the expiration. If the SA expires in the middle of processing a packet, the system finishes processing that packet.
- The actual user SA lifetime may not equal the value configured in the system.
- There are both global and tunnel-specific lifetime parameters. If there is no tunnel-specific lifetime configured, the system uses the global lifetime. The global lifetime parameters have the following default settings:
  - > 8 hours for the time-based lifetime
  - > 100 MB for the traffic-based lifetime
- Lifetime parameters are valid only for user SAs established via IKE. Manually configured user SAs ignore this parameter.

You can set a lifetime for all SAs on a specific tunnel, and you can set a global lifetime.

- To set the tunnel lifetime, use the **tunnel lifetime** command.
- To set the global (default) lifetime, use the **ipsec lifetime** command.

### Inbound and Outbound SAs

SA parameters are the actual session parameters used to secure a specific data flow associated with a specific secure IP interface.

- For manual secure IP interfaces, the system administrator sets SA parameters. Manually setting SA parameters allows provisioning of IP security to destinations that do not support SA negotiation via IKE.
- For signaled secure IP interfaces, the two security gateway peers negotiate SA parameters; the system administrator is not allowed to set any of the parameters. In fact, for some of these parameters, such as session keys, the system administrator is not granted even read-access.

Similarly to IPSec SA, SA parameters are unidirectional. Therefore, for a two-way data flow, two SAs need to be established—one for inbound traffic and another for outbound traffic. For each direction, SA parameters must be set for each transform associated with a secure IP interface. Therefore, two sets of SA parameters exist for each secure IP interface, one being the inbound SA parameters and the other the outbound SA parameters.

The following parameters form each set of SA parameters:

- SPI – A unique identifier for the SA used when securing this flow. An SPI is unique for a given destination IP address and protocol tuple. The destination IP address is either the remote secure IP interface endpoint for the outbound direction or the local secure IP interface endpoint for the inbound direction.
- Encapsulation – The encapsulation options include both an encapsulating protocol and an encapsulating mode. The protocol can be either ESP or AH. The mode is tunnel mode.
- Transforms – The allowed transforms for given SA parameters depend on its encapsulation protocol. See *Transform Sets* for more information.

- Keys – The session key used for the respective SA transform. The key length depends on the SA transform to which it applies, and is as follows:
  - > DES – 8 bytes
  - > 3DES – 24 bytes
  - > MD5 – 16 bytes
  - > SHA – 20 bytes

#### Transform Sets

Transform sets are composed of security parameters that provide a required security level to a particular data flow. Transform sets are used during user SA negotiation to find common agreement between the local and the remote security gateway on how to protect that specific data flow.

A transform set includes encapsulation protocols and transforms; for example, encryption/decryption/authentication algorithms. These parameters are grouped to specify the acceptable protection for a given data flow. Many transform sets are supported, since different traffic requires distinct security levels.

A secure IP tunnel is associated with one transform set. Multiple secure IP tunnels can refer to the same transform set.

Changing existing transform sets affects only future user SA negotiations. User SAs that are already established remain valid and do not use the changed transform set until they are renegotiated.

For manually configured secure IP tunnels, the associated transform set must contain a single transform option.

**Encapsulation Protocols** Both the AH and ESP protocols are supported. See supported transforms in Table 10-4.

- AH provides authentication.
- ESP provides data confidentiality and antireplay functions. ESP can also provide data authentication, although, in this implementation, ESP does not cover the outer IP header.

**Encapsulation Modes** IPSec supports two encapsulation modes—tunnel mode and transport mode. Currently the ERX system supports only tunnel mode. Tunnel mode creates a second IP header in the packet and uses both the local and remote security gateway addresses as source and destination IP addresses.

**Supported Transforms** Table 10-4 describes the supported transforms.

**Table 10-4** Supported transforms

Transform	Description
AH-MD5	IPSec performs AH protocol encapsulation using the MD5 hash function with HMAC message authentication.
AH-SHA	IPSec performs AH protocol encapsulation using the SHA-1 hash function with HMAC message authentication. SHA-1 is considered stronger than MD5.
ESP-MD5	IPSec performs ESP protocol encapsulation using the MD5 hash function with HMAC message authentication.
ESP-SHA	IPSec performs ESP protocol encapsulation using the SHA-1 hash function with HMAC message authentication. SHA-1 is considered stronger than MD5.
ESP-DES	IPSec performs ESP protocol encapsulation using the DES encryption algorithm. DES uses a 56-bit symmetric key and is considered a weak (breakable) encryption algorithm.
ESP-3DES	IPSec performs ESP protocol encapsulation using the 3DES encryption algorithm. 3DES uses a 168-bit symmetric encryption key and is widely accepted as a strong encryption algorithm. Export control issues apply to products that ship from the USA with 3DES.
ESP-DES-MD5	Combination of ESP-MD5 and ESP-DES transforms.
ESP-DES-SHA	Combination of ESP-SHA and ESP-DES transforms.
ESP-3DES-MD5	Combination of ESP-MD5 and ESP-3DES transforms.
ESP-3DES-SHA	Combination of ESP-SHA and ESP-3DES transforms.

Table 10-5 shows the security functions achieved with the supported transforms, and provides a view of which combinations can be used, depending on security requirements.

**Table 10-5** Combinations of supported security transforms

Security Type	Supported Transform Combinations
Data authentication only	AH-HMAC-MD5 AH-HMAC-SHA ESP-HMAC-MD5 ESP-HMAC-SHA
Data confidentiality only	ESP-DES ESP-3DES
Data authentication and confidentiality	ESP-DES-MD5 ESP-DES-SHA ESP-3DES-MD5 ESP-3DES-SHA

Note that the ERX IPSec service line module does not support both the ESP and AH encapsulation modes concurrently on the same secure tunnel.

**Negotiating Transforms** Inside a transform set, IPSec transforms are numbered in a priority sequence.

- During negotiation as an initiator of the user SA, the system uses transform number one first. If the remote system does not agree on the transform, the system then tries number two, and so on. If both end systems do not agree on a transform, the user SA fails and the secure IP tunnel is not established.
- During negotiation as a responder, the system compares the proposed transform from the remote end against each transform in the transform set. If there is no match, the system provides a negative answer to the remote end, which can either try another transform or give up. If no match is found, the secure IP tunnel is not established.

## *Other Security Features*

### IP Security Policies

The ERX system does not support a system-wide security policy database (SPD). Instead, the system takes advantage of routing to forward traffic to and from a secure tunnel. The system still applies IPSec selectors to traffic going into or coming out of a secure tunnel so that no unwanted traffic is allowed inside the tunnel. Supported selectors include IP addresses, subnets, and IP address ranges.

### ESP Processing

The system supports both the encryption and authentication functions of ESP encapsulation as defined in RFC 2406. Specifically, the system supports:

- DES and 3DES encryption algorithms
- The HMAC-SHA and HMAC-MD5 authentication algorithms
- ESP security options on a per-tunnel (per-SA) basis
- Tunnel mode

### AH Processing

The system supports AH encapsulation as defined in RFC 2402. Specifically, the system supports:

- HMAC-SHA and HMAC-MD5 authentication algorithms
- AH authentication options on a per-tunnel (per-SA) basis
- Tunnel mode

### IPSec Maximums Supported

Refer to the *ERX Release Notes, Appendix A, System Maximums* corresponding to your software release for information on maximum values.

## IKE Overview

---

The IKE suite of protocols allows a pair of security gateways to:

- Dynamically establish a secure tunnel over which the security gateways can exchange tunnel and key information.
- Set up user-level tunnels or SAs, including tunnel attribute negotiations and key management. These tunnels can also be refreshed and terminated on top of the same secure channel.

IKE is based on the Oakley and Skeme key determination protocols and the ISAKMP framework for key exchange and security association establishment. IKE provides:

- Automatic key refreshing on configurable timeout.
- Support for public key infrastructure (PKI) authentication systems.
- Antireplay defense.

IKE is layered on UDP and uses UDP port 500 to exchange IKE information between the security gateways. Therefore, UDP port 500 packets must be permitted on any IP interface involved in connecting a security gateway peer.

The following sections expand on the IKE functionality available for the system.

### *Main Mode and Aggressive Mode*

IKE phase 1 negotiations are used to negotiate keys. The two IKE peers then use these keys to communicate securely during phase 2 negotiations. IKE uses several modes for phase 1 negotiations: main mode, aggressive mode, and quick mode. You can use main mode or aggressive mode to establish the initial secure tunnels between two security gateways. After that, quick mode can be used to establish and refresh user-level SAs.

The choice of main or aggressive mode is a matter of tradeoffs. Some of the characteristics of the two modes are:

- Main mode
  - > Protects the identities of the peers during negotiations and is therefore more secure.
  - > Allows greater proposal flexibility than aggressive mode.
  - > Is more time consuming than aggressive mode because more messages are exchanged between peers.

- Aggressive mode
  - > Is faster than main mode.
  - > Exposes identities of the peers to eavesdropping, making it less secure than main mode.

### Aggressive Mode Negotiations

During aggressive mode phase 1 negotiations, the ERX system behaves as follows:

- When the ERX system is the initiator, the system searches all policy rules to find those that allow aggressive mode. The system then selects the rule with highest priority and uses the rule to initiate phase 1 negotiation. If there are no policy rules with aggressive mode allowed, the system selects the highest-priority rule that allows main mode.
- When the ERX system is the responder, the negotiation depends on what the initiator proposes, as well as what is configured in the policy rules.

Table 10-6 outlines the possible combinations of initiator proposals and policy rules. As shown, allowing aggressive mode in a policy rule allows negotiation to take place no matter what the initiator requests.

**Table 10-6** Initiator Proposals and Policy Rules

Initiator Requests	Responder Policy Rule	Match
Aggressive mode	Aggressive allowed	Yes
Aggressive mode	Aggressive not allowed	No
Main mode	Aggressive allowed	Yes
Main mode	Aggressive not allowed	Yes

The system responds to phase 1 negotiations with the highest priority policy rule that matches the initiator. A match means that all parameters, including the exchange type, match.

### *IKE Policies*

An IKE policy defines a combination of security parameters to be used during the IKE SA negotiation. IKE policies are configured on both security gateway peers, and there must be at least one policy on the local peer that matches a policy on the remote peer. Failing that, the two peers will not be able to successfully negotiate the IKE SA, and no data flow will be possible.

IKE policies are global to the system. Every IPSec service line module on a system uses the same set of policies when negotiating IKE SAs. The agreed-on IKE SA between the local system and a remote security gateway may vary, because it depends on the IKE policies used by each remote peer. However, the initial set of IKE policies the system uses is always the same and independent of which peer the system is negotiating with.

During negotiation, the system may skip IKE policies that require parameters that are not configured for the remote security gateway with which the IKE SA is being negotiated.

You can define up to ten IKE policies, with each policy having a different combination of security parameters. A default IKE policy that contains default values for every policy parameter is available. This policy is used only when IKE policies are not configured and IKE is required.

The following sections describe each of the parameters contained in an IKE policy.

#### Priority

Priority allows better (more secure) policies to be given preference during the negotiation process. The fact still remains that every IKE policy is considered secure enough to secure the IKE SA flow.

During IKE negotiation all policies are scanned, one at a time, starting from the highest-priority policy and ending with the lowest-priority policy. The first policy that the peer security gateway accepts is used for that IKE session. This procedure is repeated for every IKE session that needs to be established.

#### Encryption

A specific encryption transform can be applied to an IKE policy. The supported encryption algorithms are:

- DES
- 3DES

#### Hash Function

A specific hash function can be specified to an IKE policy. The supported ones are:

- MD5
- SHA-1

IKE also uses an authentication algorithm during IKE exchanges. This authentication algorithm is automatically set to the HMAC version of the specified hash algorithm. Therefore, you cannot have the hash function set to MD5 and authentication algorithm set to HMAC-SHA.

#### Authentication Mode

As part of the IKE protocol, one security gateway needs to authenticate the other security gateway to make sure that the IKE SA is established with the intended party. Currently, the system supports preshared keys as the authentication method.

With preshared key authentication mode, the same secret string (similar to a password) must be configured on both security gateways before the gateways can authenticate each other. It is not advisable to share a preshared key among multiple pairs of security gateways, because it reduces the key's security level.

The system allows preshared keys to be up to 256 characters composed of any ASCII alphanumeric character, except spaces.

#### Diffie-Hellman Group

An IKE policy must specify which Diffie-Hellmann group is used during the symmetrical key generation phase of IKE. The following Diffie-Hellmann groups are supported:

- Group 1 (768-bit)
- Group 2 (1024-bit)
- Group 5 (1536-bit)

#### Lifetime

Like a user SA, an IKE SA should not last indefinitely. Therefore, the system allows you to specify a lifetime parameter for an IKE policy. The timer for the lifetime parameter begins when the IKE SA is established using IKE. One minute before the IKE SA lifetime runs out, the system renegotiates the IKE SA, and new key material is established. By negotiating the SA before the lifetime expires, there is no interruption of processing the user SA.

## *IKE SA Negotiation*

As the initiator of an IKE SA, the system sends its IKE policies to the remote peer. If the peer has an IKE policy that matches the encryption, hash, authentication method, and Diffie-Hellmann group settings, the peer returns the matching policy. The peers use the lesser lifetime setting as the IKE SA lifetime. If no match is found, the IKE SA fails, and a log alarm is generated.

As the responder of an IKE negotiation, the system receives all IKE policies from a remote security gateway. The system then scans its own list of IKE policies to check whether a match exists, starting from the highest priority. If it finds a match, that policy is successfully negotiated. Again, the lifetime is negotiated to the lesser of the two lifetimes, and failures are logged.

## Configuration Tasks

---

This sections shows the steps to configure IPsec parameters, create an IPsec tunnel, and define an ISAKMP/IKE policy. The next section contains configuration examples.

### *Configuring IPsec Parameters*

To configure IPsec:

- 1 For each endpoint, create a transform set that provides the desired encryption and authentication.

```
host1(config)#ipsec transform-set customerAprotection  
    esp-3des-hmac-sha  
host1(config)#ipsec transform-set customerBprotection  
    ah-hmac-md5
```

- 2 Add a preshared key that the systems will use to authenticate each other:

```
host1(config)#ipsec key manual pre-share 5.2.0.1  
host1(config-manual-key)#key customerASecret  
host1(config-manual-key)#exit
```

- 3 Define the local endpoint used for ISAKMP/IKE negotiations for all IPsec tunnels in the router.

```
host1(config)#ipsec local-endpoint 10.10.1.1  
    transport-virtual-router vr#8
```

- 4 (Optional) Set the global (default) lifetime for all SAs on the system.

```
host1(config)#ipsec lifetime kilobytes 42000000
```

### *ipsec key manual*

- Use to specify that a peer use a manual key for authentication during the tunnel establishment phase, and display the prompt that lets you enter the manual key. To enter a key, use the **key** command.
- Specify the peer using its IP address.
- You must enter this command in the virtual router context where the IP address of the peer is defined.
- Example

```
host1(config)#ipsec key manual pre-share 10.10.1.1
host1(config-manual-key)#
```
- Use the **no** version to delete a manually configured key from the system.

### *ipsec lifetime*

- Use to set the global (default) lifetime in seconds and/or volume of traffic. The IPSec lifetime applies to tunnels that do not have a tunnel lifetime defined. When either limit is reached, the SA is renegotiated.
- To set a lifetime for all SAs on a tunnel, use the **tunnel lifetime** command.
- To set a lifetime for a specific SA, use the **lifetime** command.
- Example 1

```
host1(config)#ipsec lifetime kilobytes 42000000
```
- Example 2

```
host1(config)#ipsec lifetime seconds 8600
```
- Use the **no** version to restore the default values of 4294967295 kilobytes and 28800 seconds (8 hours).

### *ipsec local-endpoint*

- Use to define a default local endpoint for ISAKMP/IKE negotiations and all IPSec tunnels for a transport virtual router.
- You must specify the IP address used as the local endpoint and the transport virtual router on which the IP address is defined.
- Example

```
host1(config)#ipsec local-endpoint 10.10.1.1
transport-virtual-router VR#8
```
- Use the **no** version to delete a local endpoint. You cannot remove an endpoint if a tunnel is referencing the endpoint.

### *ipsec transform-set*

- Use to create a transform set. Each transform in a set provides a different combination of data authentication and confidentiality.
- Transform sets used for manually configured tunnels can have one transform.
- Transform sets used for signaled tunnels can have up to six transforms. The actual transform used on the tunnel is negotiated with the peer. Transforms are numbered in a priority sequence in the order in which you enter them.

- Example
 

```
host1(config)#ipsec transform-set espSet esp-3des-hmac-md5
      esp-3des-null-auth
```
- Use the **no** version to delete a transform set. You cannot remove a transform set if a tunnel is referencing the transform set.

### **key**

- Use to enter a manual preshared key.
- Example
 

```
host1(config-manual-key)#key dj5fe23owi8er49fdsa
```
- There is no **no** version. To delete a key, use the **no** version of the **ipsec key manual** command.

### *Creating an IPsec Tunnel*

To create an IPsec tunnel:

- 1 Enter virtual router mode. Specify the VR that contains the source and destination addresses assigned to the tunnel interface.
 

```
host1(config)#virtual-router vrA
host1:vrA(config)#
```
- 2 Create an IPsec tunnel, and specify the transport VR.
 

```
host1:vrA(config)#interface tunnel ipsec:Aottawa2boston
      transport-virtual-router default
host1:vrA(config-if)#
```
- 3 Specify the IP address of this tunnel interface.
 

```
host1:vrA(config-if)#ip address 10.3.0.0 255.255.0.0
```
- 4 Specify the transform set that ISAKMP uses for SA negotiations.
 

```
host1:vrA(config-if)#tunnel transform-set
      customerAprotection
```
- 5 Configure the local endpoint of the tunnel.
 

```
host1:vrA(config-if)#tunnel local-identity subnet 10.1.0.0
      255.255.0.0
```
- 6 Configure the peer endpoint of the tunnel.
 

```
host1:vrA(config-if)#tunnel peer-identity subnet 10.3.0.0
      255.255.0.0
```
- 7 Specify an existing interface address that the tunnel uses as its source address.
 

```
host1:vrA(config-if)#tunnel source 5.1.0.1
```

- 8 Set the address of the remote tunnel endpoint.  

```
host1:vrA(config-if)#tunnel destination 5.3.0.1
host1:vrA(config-if)#exit
```
- 9 For manual tunnels, specify the algorithm sets and the session key used for inbound SAs and for outbound SAs.  

```
host1:vrA(config-if)#tunnel session-key-inbound
  esp-3des-hmac-md5 xj82k14OWaRqy dj3EwxNQ98z3bPw9
host1:vrA(config-if)#tunnel session-key-outbound
  esp-3des-hmac-md5 421 tM967dq3KvZ3j52r HeI38pkRn963wEg4
```
- 10 (Optional) Configure PFS on this tunnel.  

```
host1:vrA(config-if)#tunnel pfs group 5
```
- 11 (Optional) Set the tunnel type to signaled or manual. The default is signaled.  

```
host1:vrA(config-if)#tunnel signaling manual
```
- 12 (Optional) Set the renegotiation time of the SAs in use by this tunnel.  

```
host1(config-if)#tunnel lifetime seconds 48000 kilobytes
  249000
```
- 13 (Optional) Set the MTU size for the tunnel.  

```
host1(config-if)#tunnel mtu 2240
```

### ***interface tunnel***

- Use to create or configure an IPsec tunnel interface.
- To establish the tunnel on a virtual router other than the current virtual router context, use the **transport-virtual-router** keyword.
- Example

```
host1(config)#interface tunnel ipsec:jak
  transport-virtual-router tvr041
host1(config-if)#
```

- Use the **no** version to remove the tunnel.

### ***tunnel destination***

- Use to set the address of the remote tunnel endpoint.
- Example  

```
host1(config-if)#tunnel destination 10.10.11.12
```
- Use the **no** version to remove the address.

### ***tunnel lifetime***

- Use to set the renegotiation time of the SAs in use by this tunnel.
- To configure the lifetime in amount of data, use the **kilobytes** keyword.
- To configure the lifetime in number of seconds, use the **seconds** keyword.
- If you include the **seconds** keyword as the first keyword on the command line, you can also include the **kilobytes** keyword on the same line.
- Before either the volume of traffic or number of seconds limit is reached, the SA is renegotiated, which ensures that the tunnel does not go down during renegotiation.

```
host1(config-if)#tunnel lifetime seconds 48000 kilobytes  
249000
```

- Use the **no** version to set the lifetime to the default lifetime, which you define using the **ipsec lifetime** command.

### ***tunnel local-identity***

- Use to configure the local identity (selector) of the tunnel. Specify the identity using one of the following keywords:
  - › **address** – specifies an IP address as the local identity
  - › **subnet** – specifies a subnet as the local identity
  - › **range** – specifies a range of IP addresses as the local identity

- Example 1

```
host1(config-if)#tunnel local-identity range 10.10.1.1  
10.10.2.1
```

- Example 2

```
host1(config-if)#tunnel local-identity subnet 10.10.1.1  
255.255.255.0
```

- Use the **no** version to set the default identity, which is subnet 0.0.0.0/0.0.0.0

### ***tunnel mtu***

- Use to set the MTU size for the tunnel.
- Example

```
host1(config-if)#tunnel mtu 2240
```

- Use the **no** version to set the MTU to the default of 1440.

### ***tunnel peer-identity***

- Use to configure the peer identity (selector) that ISAKMP uses. Specify the identity using one of the following keywords:
  - › **address** – specifies an IP address as the peer identity
  - › **subnet** – specifies a subnet as the peer identity
  - › **range** – specifies a range of IP addresses as the peer identity

- Example 1

```
host1(config-if)#tunnel peer-identity range 10.10.1.1
10.10.2.2
```
- Example 2

```
host1(config-if)#tunnel peer-identity subnet 130.10.1.1
255.255.255.0
```
- Use the **no** version to remove the peer identity.

### ***tunnel pfs group***

- Use to configure perfect forward secrecy on this tunnel.
- Assign a Diffie-Hellman prime modulus group using one of the following keywords:
  - › **1** – 768-bit group
  - › **2** – 1024-bit group
  - › **5** – 1536-bit group
- Example

```
host1(config-if)#tunnel pfs group 5
```
- Use the **no** version to remove PFS from this tunnel.

### ***tunnel session-key-inbound***

- Use to manually configure the authentication and/or encryption algorithm sets and session keys for inbound SAs on a tunnel. You can enter this command only on tunnels that have tunnel signaling set to manual.
- To see a list of available algorithm sets, use the online Help.
- Keys are an arbitrary hexadecimal string. If the algorithm set includes:
  - › DES, create an 8-byte key
  - › 3DES, create a 24-byte key
  - › MD5, create a 16-byte key
  - › SHA, create a 20-byte key
- Example

```
host1(config-if)#tunnel session-key-inbound
esp-3des-hmac-md5 xj82k140WaQp03i7 5bv0k4hm23z6uPn9
```
- Use the **no** version to remove inbound session keys from a tunnel.

### ***tunnel session-key-outbound***

- Use to manually configure the authentication and/or encryption algorithm sets, SPI, and session keys for outbound SAs on a tunnel. You can enter this command only on tunnels that have tunnel signaling set to manual.
- To see a list of available algorithm sets, use the online Help.
- The SPI is a number from 256 to 4294967295 that identifies an SA.

- Keys are an arbitrary hexadecimal string. If the algorithm set includes:
  - › DES, create an 8-byte key
  - › 3DES, create a 24-byte key
  - › MD5, create a 16-byte key
  - › SHA, create a 20-byte key
- Example

```
host1(config-if)#tunnel session-key-outbound
    esp-3des-hmac-md5 421 tM967dq3KvZ3j52r HeI38pkRn963wEg4
```
- Use the **no** version to remove outbound session keys from a tunnel.

### ***tunnel signaling***

- Use to set the tunnel type to signaled (ISAKMP) or manual. Specify a keyword:
  - › **isakmp** – uses ISAKMP/IKE to negotiate SAs and to establish keys
  - › **manual** – specifies that security parameters and keys are configured manually
- Example

```
host1(config-if)#tunnel signaling manual
```
- Use the **no** version to restore the default, **isakmp**.

### ***tunnel source***

- Use to specify an existing interface address that will serve as the tunnel's source address.
- Example

```
host1(config-if)#tunnel source 10.10.2.8
```
- Use the **no** version to remove the tunnel source.

### ***tunnel transform-set***

- Use to specify the transform set that ISAKMP uses during SA negotiations on this tunnel. You create transform sets using the **ipsec transform-set** command.
- Example

```
host1(config-if)#tunnel transform-set espSet
```
- Use the **no** version to remove the transform set from a tunnel.

### *Defining an ISAKMP/IKE Policy*

ISAKMP/IKE policies define parameters that the system uses during ISAKMP/IKE phase 1 negotiation.

To create an ISAKMP/IKE policy:

```
host1(config)#ipsec isakmp-policy-rule 3
host1(config-isakmp-policy)#
```

You can then set the following parameters, or use the default settings:

- Allow aggressive mode negotiation.

```
host1(config-isakmp-policy)#aggressive-mode
```

- Specify the authentication method.

```
host1(config-isakmp-policy)#authentication pre-share
```

- Specify the encryption algorithm.

```
host1(config-isakmp-policy)#encryption 3des
```

- Assign a Diffie-Hellman group.

```
host1(config-isakmp-policy)#group 5
```

- Set the hash algorithm.

```
host1(config-isakmp-policy)#hash md5
```

- Specify the lifetime of IKE SAs created using this policy.

```
host1(config-isakmp-policy)#lifetime 360
```

### ***aggressive-mode***

- Use to allow aggressive mode negotiation for the tunnel.
- If you allow aggressive mode negotiation, the tunnel proposes aggressive mode to the peer in connections that the policy initiates. If the peer rejects the negotiation because of the mode, the tunnel begins a main mode negotiation.
- If the peer initiates a negotiation, the tunnel accepts the negotiation if the mode matches this policy.
- Example

```
host1(config-isakmp-policy)#aggressive-mode
```

- Use the **no** version to set the negotiation mode to main mode.

### ***authentication***

- Use to specify the authentication method the system uses in the IKE policy. Currently, the only method supported is preshared keys.
- Example

```
host1(config-isakmp-policy)#authentication pre-share
```

- Use the **no** version to restore the default, preshared keys.

### ***encryption***

- Use to specify one of the following encryption algorithms to use in the IKE policy:
  - › **3des** – 168-bit 3DES-CBC
  - › **des** – 56-bit DES-CBC

- Example  
`host1(config-isakmp-policy)#encryption 3des`
- Use the **no** version to restore the default encryption algorithm, 3DES.

### **group**

- Use to assign a Diffie-Hellman group to the IKE policy. Specify:
  - › **1** – 768-bit group
  - › **2** – 1024-bit group
  - › **5** – 1536-bit group
- Example  
`host1(config-isakmp-policy)#group 5`
- Use the **no** version to restore the default, 1024-bit group.

### **hash**

- Use to set the hash algorithm for the IKE policy:
  - › **md5** – MD5 (HMAC variant)
  - › **sha** – SHA-1 (HMAC variant)
- Example  
`host1(config-isakmp-policy)#hash md5`
- Use the **no** version to restore the default, **sha**.

### **ipsec isakmp-policy-rule**

- Use to define an ISAKMP/IKE policy.
- When you enter the command, you include a number that identifies the policy and assigns a priority to the policy. You can number policies from 1 to 10000, with 1 having the highest priority.
- You can add up to 10 ISAKMP/IKE policies per system.
- Example  
`host1(config)#ipsec isakmp-policy-rule 3`  
`host1(config-isakmp-policy)#`
- Use the **no** version to remove policies. If you do not include a priority number with the **no** version, all policies are removed.

### **lifetime**

- Use to specify the lifetime of IKE SAs.
- The range is 7200 to 860400 seconds.  
`host1(config-isakmp-policy)#lifetime 360`
- Use the **no** version to reset the SA lifetime to the default, 28800 seconds.

## Refreshing SAs

To refresh ISAKMP/IKE or IPsec SAs:

```
host1(config)#ipsec clear sa tunnel ipsec:Aottawa2boca  
phase 2
```

### **ipsec clear sa**

- Use to refresh ISAKMP/IKE or IPsec SAs.
- To reinitialize all SAs, use the **all** keyword.
- To reinitialize SAs on a specific tunnel, use the **tunnel** keyword.
- To reinitialize SAs on tunnels that are in a specific state, use the **state** keyword.
- To specify the type of SA to be reinitialized, ISAKMP/IKE or IPSEC, use the **phase** keyword.
- Example  

```
host1(config)#ipsec clear sa all phase 2
```
- There is no **no** version.

## Configuration Examples

---

This section contains examples of two IPsec applications. The first example shows a customer who replaces a leased line network with an IPsec network that allows the company to connect its corporate locations over the Internet. The second example provides leased line replacement to two customers who use address schemes in the same range.

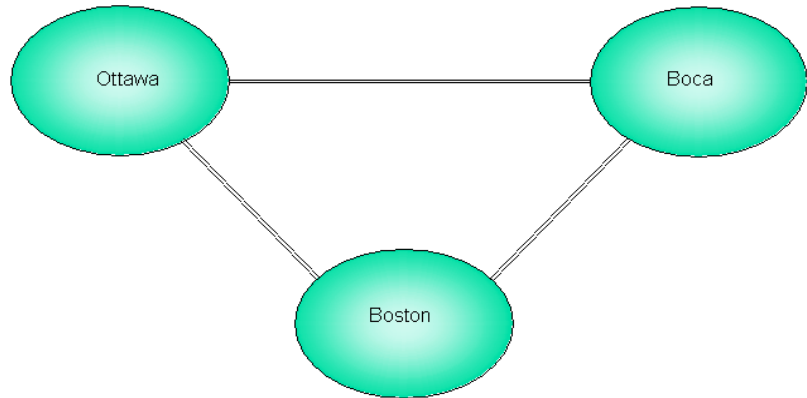
### *Configuration Notes*

Both the local and remote identities shown in these examples serve two purposes:

- They identify multiple IPsec tunnels between the same endpoints.
- They filter traffic going into and coming out of the tunnels so that it is within the specified range. If the configuration requires that only one IPsec tunnel exists between two endpoints and no traffic filtering is required, you can omit the **tunnel local-identity** and **tunnel peer-identity** commands.

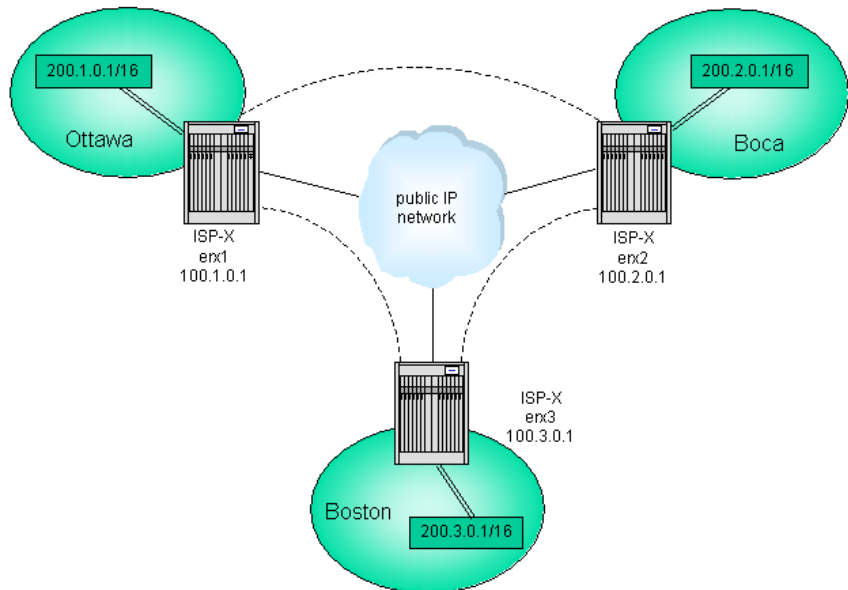
*Example 1*

In Figure 10-4 customer A is using Frame Relay to connect its corporate offices in three cities: Boston, Ottawa, and Boca.



**Figure 10-4** Customer A's corporate Frame Relay network

Customer A hires ISP-X to provide a leased line replacement over an IP infrastructure using IPSec. ISP-X can offer a replacement for long-haul Frame Relay links by creating IPSec tunnels to carry customer A's traffic securely between the sites over the public or ISP-provided IP network. This alternative will cost only a fraction of the price of the Frame Relay links. Figure 10-5 shows the connectivity scheme.



**Figure 10-5** ISP-X uses ERX systems to connect corporate offices over the Internet

To configure the connections as shown in Figure 10-5:

- 1 On each ERX system, create a protection suite that provides 3DES encryption with SHA-1 authentication on every packet.

```
erx1(config)#ipsec transform-set customerAprotection  
    esp-3des-hmac-sha
```

```
erx2(config)#ipsec transform-set customerAprotection  
    esp-3des-hmac-sha
```

```
erx3(config)#ipsec transform-set customerAprotection  
    esp-3des-hmac-sha
```

- 2 On each ERX system, create preshared keys that the three systems will use to authenticate each other:

```
erx1(config)#ipsec key manual pre-share 100.2.0.1
```

```
erx1(config-manual-key)#key customerASecret
```

```
erx1(config-manual-key)#exit
```

```
erx1(config)#ipsec key manual pre-share 100.3.0.1
```

```
erx1(config-manual-key)#key customerASecret
```

```
erx1(config-manual-key)#exit
```

```
erx2(config)#ipsec key manual pre-share 100.1.0.1
```

```
erx2(config-manual-key)#key customerASecret
```

```
erx2(config-manual-key)#exit
```

```
erx2(config)#ipsec key manual pre-share 100.3.0.1
```

```
erx2(config-manual-key)#key customerASecret
```

```
erx2(config-manual-key)#exit
```

```
erx3(config)#ipsec key manual pre-share 100.1.0.1
```

```
erx3(config-manual-key)#exit
```

```
erx3(config-manual-key)#key customerASecret
```

```
erx3(config)#ipsec key manual pre-share 100.2.0.1
```

```
erx3(config-manual-key)#key customerASecret
```

```
erx3(config-manual-key)#exit
```

- 3 On erx1 create two IPSec tunnels, one to carry customer A's traffic between Ottawa and Boston and another to carry the traffic between Ottawa and Boca:

Tunnel 1:

```
erx1(config)#interface tunnel ipsec:Aottawa2boston
```

```
erx1(config-if)#tunnel transform-set customerAprotection
```

```
erx1(config-if)#tunnel local-identity subnet 200.1.0.0  
    255.255.0.0
```

```
erx1(config-if)#tunnel peer-identity subnet 200.3.0.0  
    255.255.0.0
```

```
erx1(config-if)#tunnel source 100.1.0.1
```

```
erx1(config-if)#tunnel destination 100.3.0.1
```

```
erx1(config-if)#ip address 200.3.0.0 255.255.0.0
erx1(config-if)#exit
```

#### Tunnel 2:

```
erx1(config)#interface tunnel ipsec:Aottawa2boca
erx1(config-if)#tunnel transform-set customerAprotection
erx1(config-if)#tunnel local-identity subnet 200.1.0.0
255.255.0.0
erx1(config-if)#tunnel peer-identity subnet 200.2.0.0
255.255.0.0
erx1(config-if)#tunnel source 100.1.0.1
erx1(config-if)#tunnel destination 100.2.0.1
erx1(config-if)#ip address 200.2.0.0 255.255.0.0
erx1(config-if)#exit
```

- 4 On erx2 create two IPSec tunnels, one to carry customer A's traffic between Boca and Ottawa and another to carry the traffic between Boca and Boston:

#### Tunnel 1:

```
erx2(config)#interface tunnel ipsec:Aboca2ottawa
erx2(config-if)#tunnel transform-set customerAprotection
erx2(config-if)#tunnel local-identity subnet 200.2.0.0
255.255.0.0
erx2(config-if)#tunnel peer-identity subnet 200.1.0.0
255.255.0.0
erx2(config-if)#tunnel source 100.2.0.1
erx2(config-if)#tunnel destination 100.1.0.1
erx2(config-if)#ip address 200.1.0.0 255.255.0.0
erx2(config-if)#exit
```

#### Tunnel 2:

```
erx2(config)#interface tunnel ipsec:Aboca2boston
erx2(config-if)#tunnel transform-set customerAprotection
erx2(config-if)#tunnel local-identity subnet 200.2.0.0
255.255.0.0
erx2(config-if)#tunnel peer-identity subnet 200.3.0.0
255.255.0.0
erx2(config-if)#tunnel source 100.2.0.1
erx2(config-if)#tunnel destination 100.3.0.1
erx2(config-if)#ip address 200.3.0.0 255.255.0.0
erx2(config-if)#exit
```

- 5 Finally, on erx3 create two IPSec tunnels, one to carry customer A's traffic between Boston and Ottawa and another to carry the traffic between Boston and Boca:

### Tunnel 1:

```
erx3(config)#interface tunnel ipsec:Aboston2ottawa
erx3(config-if)#tunnel transform-set customerAprotection
erx3(config-if)#tunnel local-identity subnet 200.3.0.0
255.255.0.0
erx3(config-if)#tunnel peer-identity subnet 200.1.0.0
255.255.0.0
erx3(config-if)#tunnel source 100.3.0.1
erx3(config-if)#tunnel destination 100.1.0.1
erx3(config-if)#ip address 200.1.0.0 255.255.0.0
erx3(config-if)#exit
```

### Tunnel 2:

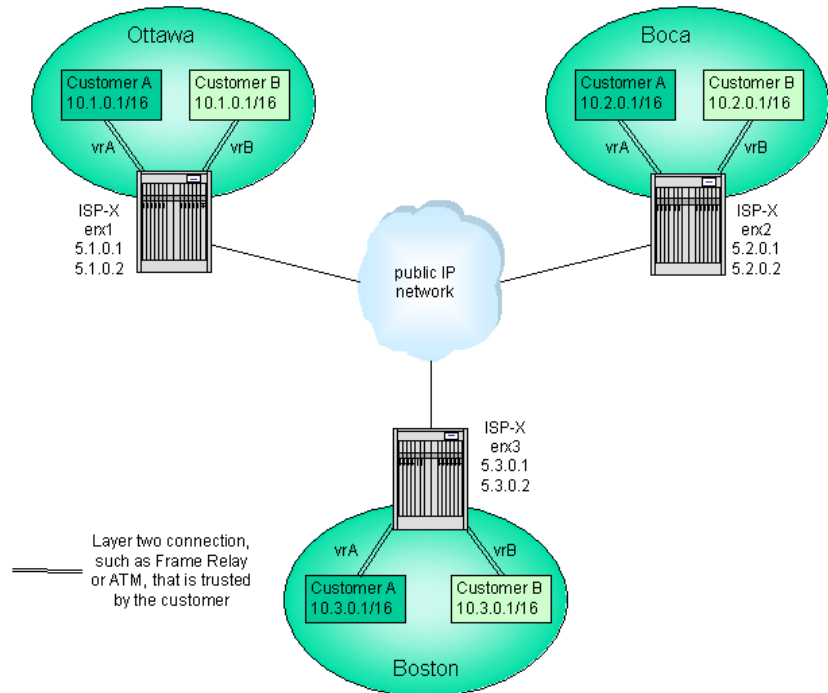
```
erx3(config)#interface tunnel ipsec:Aboston2boca
erx3(config-if)#tunnel transform-set customerAprotection
erx3(config-if)#tunnel local-identity subnet 200.3.0.0
255.255.0.0
erx3(config-if)#tunnel peer-identity subnet 200.2.0.0
255.255.0.0
erx3(config-if)#tunnel source 100.3.0.1
erx3(config-if)#tunnel destination 100.2.0.1
erx3(config-if)#ip address 200.2.0.0 255.255.0.0
erx3(config-if)#exit
```

The configuration is complete. Now customer A traffic between different cities flows through the public, or untrusted, IP network inside a tunnel, where each packet is encrypted and authenticated. Of course, this example shows the basic secure encapsulation of customer traffic over the untrusted IP network. You can add features such as key refreshing.

### Example 2

Example 2, shown in Figure 10-6, enhances the previous example by having the same ISP-X providing leased line replacement to two customers who use address schemes in the same range. There are two ways to solve scenarios in which different customers use similar IP address schemes:

- One solution is to have different transport virtual routers—a configuration similar to example 1, except that a different VR domain is possible.
- Another solution, as described in this example, simply duplicates the endpoints for the transport VR. This example assumes that the transport VR is the default VR.



**Figure 10-6** Connecting customers who use similar address schemes

To configure the connections as shown in Figure 10-6:

- 1 On each ERX system, create a protection suite that provides customer A with 3DES encryption and SHA-1 authentication, and customer B with AH authentication using MD5.

```
erx1(config)#ipsec transform-set customerAprotection
esp-3des-hmac-sha
```

```
erx1(config)#ipsec transform-set customerBprotection
ah-hmac-md5
```

```
erx2(config)#ipsec transform-set customerAprotection
esp-3des-hmac-sha
```

```
erx2(config)#ipsec transform-set customerBprotection
ah-hmac-md5
```

```
erx3(config)#ipsec transform-set customerAprotection
esp-3des-hmac-sha
```

```
erx3(config)#ipsec transform-set customerBprotection
ah-hmac-md5
```

- 2 On each ERX system, create a protection suite that the three systems will use to authenticate each other:

```
erx1(config)#ipsec key manual pre-share 5.2.0.1
erx1(config-manual-key)#key customerASecret
erx1(config-manual-key)#exit
erx1(config)#ipsec key manual pre-share 5.3.0.1
erx1(config-manual-key)#key customerASecret
erx1(config-manual-key)#exit

erx1(config)#ipsec key manual pre-share 5.2.0.2
erx1(config-manual-key)#key customerBSecret
erx1(config-manual-key)#exit
erx1(config)#ipsec key manual pre-share 5.3.0.2
erx1(config-manual-key)#key customerBSecret
erx1(config-manual-key)#exit

erx2(config)#ipsec key manual pre-share 5.1.0.1
erx2(config-manual-key)#key customerASecret
erx2(config-manual-key)#exit
erx2(config)#ipsec key manual pre-share 5.3.0.1
erx2(config-manual-key)#key customerASecret
erx2(config-manual-key)#exit

erx2(config)#ipsec key manual pre-share 5.1.0.2
erx2(config-manual-key)#key customerBSecret
erx2(config-manual-key)#exit
erx2(config)#ipsec key manual pre-share 5.3.0.2
erx2(config-manual-key)#key customerBSecret
erx2(config-manual-key)#exit

erx3(config)#ipsec key manual pre-share 5.1.0.1
erx3(config-manual-key)#key customerASecret
erx3(config-manual-key)#exit
erx3(config)#ipsec key manual pre-share 5.2.0.1
erx3(config-manual-key)#key customerASecret
erx3(config-manual-key)#exit

erx3(config)#ipsec key manual pre-share 5.1.0.2
erx3(config-manual-key)#key customerBSecret
erx3(config-manual-key)#exit
erx3(config)#ipsec key manual pre-share 5.2.0.2
erx3(config-manual-key)#key customerBSecret
erx3(config-manual-key)#exit
```

- 3 On erx1, create two IPSec tunnels, one to carry customer A's traffic and another to carry customer B's traffic. You must create each pair of tunnels in the virtual routers where the IP interfaces reaching those customers are defined. The endpoints for the tunnels should be created in the ISP default virtual router.

## Virtual router A:

```
erxl(config)#virtual-router vrA  
erxl:vrA(config)#
```

## Tunnel from Ottawa to Boston on virtual router A:

```
erxl:vrA(config)#interface tunnel ipsec:Aottawa2boston  
                  transport-virtual-router default  
erxl:vrA(config-if)#tunnel transform-set customerAprotection  
erxl:vrA(config-if)#tunnel local-identity subnet 10.1.0.0  
                  255.255.0.0  
erxl:vrA(config-if)#tunnel peer-identity subnet 10.3.0.0  
                  255.255.0.0  
erxl:vrA(config-if)#tunnel source 5.1.0.1  
erxl:vrA(config-if)#tunnel destination 5.3.0.1  
erxl:vrA(config-if)#ip address 10.3.0.0 255.255.0.0  
erxl:vrA(config-if)#exit
```

## Tunnel from Ottawa to Boca on virtual router A:

```
erxl:vrA(config)#interface tunnel ipsec:Aottawa2boca  
                  transport-virtual-router default  
erxl:vrA(config-if)#tunnel transform-set customerAprotection  
erxl:vrA(config-if)#tunnel local-identity subnet 10.1.0.0  
                  255.255.0.0  
erxl:vrA(config-if)#tunnel peer-identity subnet 10.2.0.0  
                  255.255.0.0  
erxl:vrA(config-if)#tunnel source 5.1.0.1  
erxl:vrA(config-if)#tunnel destination 5.2.0.1  
erxl:vrA(config-if)#ip address 10.2.0.0 255.255.0.0  
erxl:vrA(config-if)#exit
```

## Virtual router B:

```
erxl(config)#virtual-router vrB  
erxl:vrB(config)#
```

## Tunnel from Ottawa to Boston on virtual router B:

```
erxl:vrB(config)#interface tunnel ipsec:Bottawa2boston  
                  transport-virtual-router default  
erxl:vrB(config-if)#tunnel transform-set customerBprotection  
erxl:vrB(config-if)#tunnel local-identity subnet 10.1.0.0  
                  255.255.0.0  
erxl:vrB(config-if)#tunnel peer-identity subnet 10.3.0.0  
                  255.255.0.0  
erxl:vrB(config-if)#tunnel source 5.1.0.2  
erxl:vrB(config-if)#tunnel destination 5.3.0.2  
erxl:vrB(config-if)#ip address 10.3.0.0 255.255.0.0  
erxl:vrB(config-if)#exit
```

### Tunnel from Ottawa to Boca on virtual router B:

```
erx1:vrB(config)#interface tunnel ipsec:Bottawa2boca
transport-virtual-router default
erx1:vrB(config-if)#tunnel transform-set customerBprotection
erx1:vrB(config-if)#tunnel local-identity subnet 10.1.0.0
255.255.0.0
erx1:vrB(config-if)#tunnel peer-identity subnet 10.2.0.0
255.255.0.0
erx1:vrB(config-if)#tunnel source 5.1.0.2
erx1:vrB(config-if)#tunnel destination 5.2.0.2
erx1:vrB(config-if)#ip address 10.2.0.0 255.255.0.0
erx1:vrB(config-if)#exit
```

- 4 On erx2, create two IPSec tunnels, one to carry customer A's traffic and another to carry customer B's traffic. You must create each pair of tunnels in the virtual routers where the IP interfaces reaching those customers are defined. The endpoints for the tunnels should be created in the ISP default virtual router.

### Virtual router A:

```
erx2(config)#virtual-router vrA
erx2:vrA(config)#
```

### Tunnel from Boca to Ottawa on virtual router A:

```
erx2:vrA(config)#interface tunnel ipsec:Aboca2ottawa
transport-virtual-router default
erx2:vrA(config-if)#tunnel transform-set customerAprotection
erx2:vrA(config-if)#tunnel local-identity subnet 10.2.0.0
255.255.0.0
erx2:vrA(config-if)#tunnel peer-identity subnet 10.1.0.0
255.255.0.0
erx2:vrA(config-if)#tunnel source 5.2.0.1
erx2:vrA(config-if)#tunnel destination 5.1.0.1
erx2:vrA(config-if)#ip address 10.1.0.0 255.255.0.0
erx2:vrA(config-if)#exit
```

### Tunnel from Boca to Boston on virtual router A:

```
erx2:vrA(config)#interface tunnel ipsec:Aboca2boston
transport-virtual-router default
erx2:vrA(config-if)#tunnel transform-set customerAprotection
erx2:vrA(config-if)#tunnel local-identity subnet 10.2.0.0
255.255.0.0
erx2:vrA(config-if)#tunnel peer-identity subnet 10.3.0.0
255.255.0.0
erx2:vrA(config-if)#tunnel source 5.2.0.1
erx2:vrA(config-if)#tunnel destination 5.3.0.1
erx2:vrA(config-if)#ip address 10.3.0.0 255.255.0.0
erx2:vrA(config-if)#exit
```

## Virtual router B:

```
erx2(config)#virtual-router vrB  
erx2:vrB(config)#
```

## Tunnel from Boca to Ottawa on virtual router B:

```
erx2:vrB(config)#interface tunnel ipsec:Bboca2ottawa  
                  transport-virtual-router default  
erx2:vrB(config-if)#tunnel transform-set customerBprotection  
erx2:vrB(config-if)#tunnel local-identity subnet 10.2.0.0  
                  255.255.0.0  
erx2:vrB(config-if)#tunnel peer-identity subnet 10.1.0.0  
                  255.255.0.0  
erx2:vrB(config-if)#tunnel source 5.2.0.2  
erx2:vrB(config-if)#tunnel destination 5.1.0.2  
erx2:vrB(config-if)#ip address 10.1.0.0 255.255.0.0  
erx2:vrB(config-if)#exit
```

## Tunnel from Boca to Boston on virtual router B:

```
erx2:vrB(config)#interface tunnel ipsec:Bboca2boston  
                  transport-virtual-router default  
erx2:vrB(config-if)#tunnel transform-set customerBprotection  
erx2:vrB(config-if)#tunnel local-identity subnet 10.2.0.0  
                  255.255.0.0  
erx2:vrB(config-if)#tunnel peer-identity subnet 10.3.0.0  
                  255.255.0.0  
erx2:vrB(config-if)#tunnel source 5.2.0.2  
erx2:vrB(config-if)#tunnel destination 5.3.0.2  
erx2:vrB(config-if)#ip address 10.3.0.0 255.255.0.0  
erx2:vrB(config-if)#exit
```

- 5 Last, on erx3, create two IPsec tunnels, one to carry customer A's traffic and another to carry customer B's traffic.

## Virtual router A:

```
erx3(config)#virtual-router vrA  
erx3:vrA(config)#
```

## Tunnel from Boston to Ottawa on virtual router A:

```
erx3:vrA(config)#interface tunnel ipsec:Aboston2ottawa  
                  transport-virtual-router default  
erx3:vrA(config-if)#tunnel transform-set customerAprotection  
erx3:vrA(config-if)#tunnel local-identity subnet 10.3.0.0  
                  255.255.0.0  
erx3:vrA(config-if)#tunnel peer-identity subnet 10.1.0.0  
                  255.255.0.0  
erx3:vrA(config-if)#tunnel source 5.3.0.1  
erx3:vrA(config-if)#tunnel destination 5.1.0.1  
erx3:vrA(config-if)#ip address 10.1.0.0 255.255.0.0  
erx3:vrA(config-if)#exit
```

### Tunnel from Boston to Boca on virtual router A:

```
erx3:vrA(config)#interface tunnel ipsec:Aboston2boca
transport-virtual-router default
erx3:vrA(config-if)#tunnel transform-set customerAprotection
erx3:vrA(config-if)#tunnel local-identity subnet 10.3.0.0
255.255.0.0
erx3:vrA(config-if)#tunnel peer-identity subnet 10.2.0.0
255.255.0.0
erx3:vrA(config-if)#tunnel source 5.3.0.1
erx3:vrA(config-if)#tunnel destination 5.2.0.1
erx3:vrA(config-if)#ip address 10.1.0.0 255.255.0.0
erx3:vrA(config-if)#exit
```

### Virtual router B:

```
erx3(config)#virtual-router vrB
erx3:vrB(config)#
```

### Tunnel from Boston to Ottawa on virtual router B:

```
erx3:vrB(config)#interface tunnel ipsec:Bboston2ottawa
transport-virtual-router default
erx3:vrB(config-if)#tunnel transform-set customerBprotection
erx3:vrB(config-if)#tunnel local-identity subnet 10.3.0.0
255.255.0.0
erx3:vrB(config-if)#tunnel peer-identity subnet 10.1.0.0
255.255.0.0
erx3:vrB(config-if)#tunnel source 5.3.0.1
erx3:vrB(config-if)#tunnel destination 5.1.0.1
erx3:vrB(config-if)#ip address 10.1.0.0 255.255.0.0
erx3:vrB(config-if)#exit
```

### Tunnel from Boston to Boca on virtual router B:

```
erx3:vrB(config)#interface tunnel ipsec:Bboston2boca
transport-virtual-router default
erx3:vrB(config-if)#tunnel transform-set customerBprotection
erx3:vrB(config-if)#tunnel local-identity subnet 10.3.0.0
255.255.0.0
erx3:vrB(config-if)#tunnel peer-identity subnet 10.2.0.0
255.255.0.0
erx3:vrB(config-if)#tunnel source 5.3.0.1
erx3:vrB(config-if)#tunnel destination 5.2.0.1
erx3:vrB(config-if)#ip address 10.2.0.0 255.255.0.0
erx3:vrB(config-if)#exit
```

The configuration is complete. Customer A's traffic and customer B's traffic can flow through the public, or untrusted, IP network inside a tunnel, where each packet is encrypted and authenticated.

## Monitoring IPsec

---

This section describes the **show** commands that you can use to view your IPsec configuration and to monitor IPsec tunnels and statistics.

### **show ike policy-rule**

- Use to display the configuration of IKE phase 1 policy rules.
- Field descriptions
  - › Protection suite priority – priority number assigned to the policy rule
  - › encryption algorithm – encryption algorithm used in the IKE policy: des, 3des
  - › hash algorithm – hash algorithm used in the IKE policy: SHA, MD5
  - › authentication method – authentication method used in the IKE policy: RSA signature, RSA encrypted nonce, DSS signature, preshared keys
  - › Diffie-Hellman group – size of the Diffie-Hellman group: 768-bit, 1024-bit, 1536-bit
  - › lifetime – lifetime of SAs created with this policy: 60 to 86400 seconds
  - › aggressive mode – allowed or not allowed
- Example

```
host1#show ike policy-rule
```

```
IKE Policy Rules:
```

```
Protection suite priority: 5
```

```
encryption algorithm :3DES Triple Data Encryption Standard(168 bit keys)
hash algorithm       :SHA Secure Hash Standard
authentication method:Pre Shared Keys
Diffie-Hellman group :5 (1536 bit)
lifetime             :7200 seconds
aggressive mode     :Not Allowed
```

```
Protection suite priority: 6
```

```
encryption algorithm :3DES Triple Data Encryption Standard(168 bit keys)
hash algorithm       :SHA Secure Hash Standard
authentication method:Pre Shared Keys
Diffie-Hellman group :2 (1024 bit)
lifetime             :28800 seconds
aggressive mode     :Not Allowed
```

### **show ike sa**

- Use to display IKE phase 1 SAs running on the system.
- Field descriptions
  - › Source – local IP address of phase 1 negotiation
  - › Destination – remote IP address of phase 1 negotiation

- › Time(s) – time remaining in phase 1 lifetime, in seconds
- › State – current state of the phase 1 negotiation. Corresponds to the messaging state in the main mode and aggressive mode negotiations. The number in brackets corresponds to the state number. Possible states are:
  - MM\_SA\_I Sent SA – initiator has sent initial main mode SA payload to the responder
  - MM\_SA\_R Sent SA – responder has sent a response to the initial main mode SA
  - MM\_KE\_I Sent KE – initiator has sent initial main mode key exchange to the responder
  - MM\_KE\_R Sent KE – responder has sent a response to the key exchange
  - MM\_FINAL\_I Sent final packet – initiator has sent the final packet in the main mode negotiation
  - MM\_FINAL\_R Sent final packet – responder has finished main mode negotiation
  - MM\_DONE\_I Done – initiator has finished main mode negotiation
  - AM\_SA\_I Sent SA, KE – initiator has sent initial aggressive mode SA payload and key exchange to the responder
  - AM\_SA\_R Sent SA, KE – responder has sent aggressive mode SA payload and key exchange to the initiator
  - AM\_FINAL\_I Sent final packet – initiator has finished aggressive mode negotiation
  - AM\_DONE\_R Done – responder has finished aggressive mode negotiation

- Example

```
host1#show ike sa
IKE Phase 1 SA's:
Source          Destination    Time(s) State
10.13.0.99      10.13.0.100   12585   MM_DONE_I Done(9)
10.13.1.99      10.13.1.100   12584   MM_DONE_I Done(9)
10.13.2.99      10.13.2.100   12584   MM_DONE_I Done(9)
10.13.3.99      10.13.3.100   12591   MM_DONE_I Done(9)
10.13.4.99      10.13.4.100   12598   MM_DONE_I Done(9)
10.13.5.99      10.13.5.100   12632   MM_DONE_I Done(9)
10.13.6.99      10.13.6.100   12658   MM_DONE_I Done(9)
10.13.7.99      10.13.7.100   12690   MM_DONE_I Done(9)
10.13.8.99      10.13.8.100   12716   MM_DONE_I Done(9)
```

### ***show ipsec lifetime***

- Use to display the configured IPsec default lifetime.
- Example

```
host1#show ipsec lifetime
Default lifetime in seconds is '7200'.
Default lifetime in kilobytes is '4294967295'.
```

**show ipsec local-endpoint**

- Use to display the address and transport virtual router of local endpoints.
- To display the local endpoint of a specific transport virtual router, include the virtual router name.
- Example

```
host1#show ipsec local-endpoint transport-virtual-router
default
Local endpoint for transport-virtual-router default is
'0.0.0.0'.
```

**show ipsec transform-set**

- Use to display transform sets configured on the system.
- To display a specific transform set, include the transform set name.
- Field descriptions
  - › Transform-set – displays the transforms in the transform set
- Example 1

```
host1#show ipsec transform-set
Transform-set: Highest security = {esp-3des-hmac-sha }.
Transform-set: transform-esp-3des-hmac-sha =
{esp-3des-hmac-sha }.
```

- Example 2

```
host1#show ipsec transform-set transform-esp-3des-hmac-sha
Transform-set: transform-esp-3des-hmac-sha =
{esp-3des-hmac-sha }.
```

**show ipsec tunnel detail**

- Use to display the running configuration and statistics for each tunnel.
- Field descriptions
  - › IPSEC tunnel – name and state of tunnel for which information is displayed
  - › Tunnel operational configuration – configuration running on the tunnel
    - Tunnel type – manual, signaled
    - Tunnel mtu – MTU size of the tunnel
    - Tunnel localEndpoint – IP address of local tunnel endpoint
    - Tunnel destination address – IP address of tunnel destination
    - Tunnel transport virtual router – name of transport virtual router over which tunnel runs
    - Tunnel transform set – tunnel transform set in use on this tunnel
    - Tunnel local identity – IP address of local endpoint identity that ISAKMP uses
    - Tunnel peer identity – IP address of peer endpoint identity that ISAKMP uses
    - Tunnel inbound spi/SA – SPI and SA in use on traffic received from the tunnel
    - Tunnel outbound spi/SA – SPI and SA in use on traffic sent to the tunnel
    - Tunnel lifetime seconds – configured lifetime in seconds

- Tunnel lifetime kilobytes – configured lifetime in kilobytes
- Tunnel pfs – PFS group in use on the tunnel: 0 (PFS is not in use), 1 (768-bit group), 2 (1024-bit group), 5 (1536-bit group)
- Tunnel administrative state – Up, Down
- › Tunnel Statistics – displays statistics on traffic received on and sent from this tunnel
  - InUserPackets – number of user packets received
  - InUserOctets – number of octets received from user packets
  - InAccPackets – number of encapsulated packets received
  - InAccOctets – number of octets received in encapsulated packets
  - InAuthErrors – number of authentication errors received
  - InReplayErrors – number of replay errors in received traffic
  - InPolicyErrors – number of policy errors in received traffic
  - InOtherRxErrors – number of packets received that have errors other than those listed above
  - InDecryptErrors – number of decryption errors in received traffic
  - InPadErrors – number of packets received that had invalid values after the packet was decrypted
  - OutUserPackets – number of user packets sent
  - OutUserOctets – number of octets sent in user packets
  - OutAccPackets – number of encapsulated packets sent
  - OutAccOctets – number of octets sent in encapsulated packets
  - OutPolicyErrors – number of packets arriving at tunnel for encapsulation that do not meet specified tunnel identifier (selector)
  - OutOtherTxErrors – number of outbound packets that have errors other than those listed above
- Example

```
host1#show ipsec tunnel detail
IPSEC tunnel s011e3d0 is up
Tunnel operational configuration:
  Tunnel type is signaled
  Tunnel mtu is 1440
  Tunnel localEndpoint is 10.255.1.13
  Tunnel destination address is 10.255.1.14
  Tunnel transport virtual router is vr00
  Tunnel transform set is transform-esp-3des-hmac-sha
  Tunnel local identity is ipAddress: 10.0.10.3
  Tunnel peer identity is ipAddress: 10.0.11.3
  Tunnel inbound spi is 0x13c00201, SA is esp-3des-hmac-sha.
  Tunnel outbound spi is 0x12b90209, SA is
  esp-3des-hmac-sha.
  Tunnel lifetime seconds is 7200
  Tunnel lifetime kilobytes is 102400
  Tunnel pfs is group 5
```

```

Tunnel administrative state is Up
Tunnel Statistics :
  InUserPackets      297
  InUserOcets        306232
  InAccPackets       297
  InAccOcets         322864
  InAuthErrors       0
  InReplayErrors     0
  InPolicyErrors     0
  InOtherRxErrors    0
  InDecryptErrors    0
  InPadErrors        0

  OutUserPackets     316
  OutUserOctes       317424
  OutAccPackets      316
  OutAccOcets        335120
  OutPolicyErrors    0
  OutOtherTxErrors   0

```

**show ipsec tunnel summary**

- Use to display a summary of all tunnels configured on the system.
- Field descriptions
  - › Total number of ipsec interface – number of tunnels configured on the system
  - › Administrative status – number of tunnels with an administrative status of enabled and disabled
  - › Operational status – number of tunnels with an operational status of up, down, lower layer down, not present
- Example

```

host1#show ipsec tunnel summary
Total number of ipsec interface is 40
Administrative status   enabled   disabled
                       40          0

Operational status   up        down        lower-down  not-present
                    40         0          0           0

```

**show ipsec tunnel virtual-router**

- Use to display the status of tunnels configured on a virtual router.
- To display only tunnels that are in a specific state, use the **state** keyword.
- To display tunnels that are using a particular IP address, use the **ip** keyword.
- Field descriptions
  - › For a description of fields, see the **show ipsec tunnel detail** command.

- Example

```
host1#show ipsec tunnel virtual-router default ip
10.255.1.13
IPSEC tunnel s011e3d0 is up
IPSEC tunnel s011e3d1 is up
IPSEC tunnel s012e3d0 is up
IPSEC tunnel s012e3d1 is up
IPSEC tunnel s013e3d0 is up
IPSEC tunnel s014e3d0 is up
IPSEC tunnel s014e3d1 is up
IPSEC tunnel s015e3d0 is up
```

