Chapter 6
Configuring IPSec

This chapter describes Internet Protocol Security (IPSec) capabilities of the ERX routers. It contains the following sections:

Overview

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

- Encapsulating protocols, including authentication (AH) and Encapsulating Security Payload (ESP), to provide security on specified packets
- The Internet Security Association and Key Management Protocol/Internet Key Exchange (ISAKMP/IKE) protocol suite to provide automatic negotiation of security associations, including session keys

IPSec Terms and Acronyms

Table 9 describes terms and abbreviations that are used in this discussion of IPSec.

Table 9: IPSec Terms and Abbreviations

Term or Abbreviation	Description
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
DPD	Dead peer detection, which enables router to detect when communication to remote peer has been disconnected. Also known as IKE keepalive.
DSS	Digital Signature Standard authentication algorithm
ESP	Encapsulating Security Payload, which provides data integrity, data confidentiality and, optionally, sender's authentication
FQDN	Fully qualified domain name, which consists of the hostname and domain name for a specific system
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, 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	Internet Protocol 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 1.
MD*x*	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, the clear traffic forwarded to the interface (either by policy or by routing) that is typically 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.
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

Platform Considerations

For information about modules that support IPSec on ERX-14xx models, ERX-7xx models, and the ERX-310 router:

- See *ERX Module Guide, Table 1, Module Combinations* for detailed module specifications.
- See *ERX Module Guide, Appendix A, Module Protocol Support* for information about the modules that support IPSec.

NOTE: The E120 router and the E320 router do not support configuration of IPSec.

References

For information about IPSec, see the following RFCs:

- RFC 768—User Datagram Protocol (August 1980)
- 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 3706—A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers (February 2004)

For information about using digital certificates, see *Chapter 9, Configuring Digital Certificates*.

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 encapsulation protocols:

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

Both protocols are defined with two modes of operation:

- Tunnel mode completely encapsulates the original packet within another IP header.
- Transport mode keeps the original header and does not add the extra IP header.

Secure IP Interfaces

Secure IP interfaces are virtual IP interfaces that you can configure 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.

Secure IP interfaces connect the router 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 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.

RFC 2401 Compliance

RFC 2401 states that a security policy database (SPD) must exist for *each* physical interface in the router, and an administrator must configure these SPDs to determine which traffic must be IPSec-protected, not IPSec-protected, or denied. The ERX router does not support a systemwide SPD. Instead, the router takes advantage of routing policies that are applied to physical interfaces to describe which traffic to forward to a single IPSec tunnel, which traffic to discard, and so on. The router also applies IPSec selectors to traffic going into or coming out of a secure tunnel so that unwanted traffic is not allowed inside the tunnel. Supported selectors include IP addresses, subnets, and IP address ranges. An implementation that strictly follows RFC 2401 requires a separate IPSec tunnel for each SPD entry.

IPSec Protocol Stack

Figure 12 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 12: IPSec Tunneling Stack

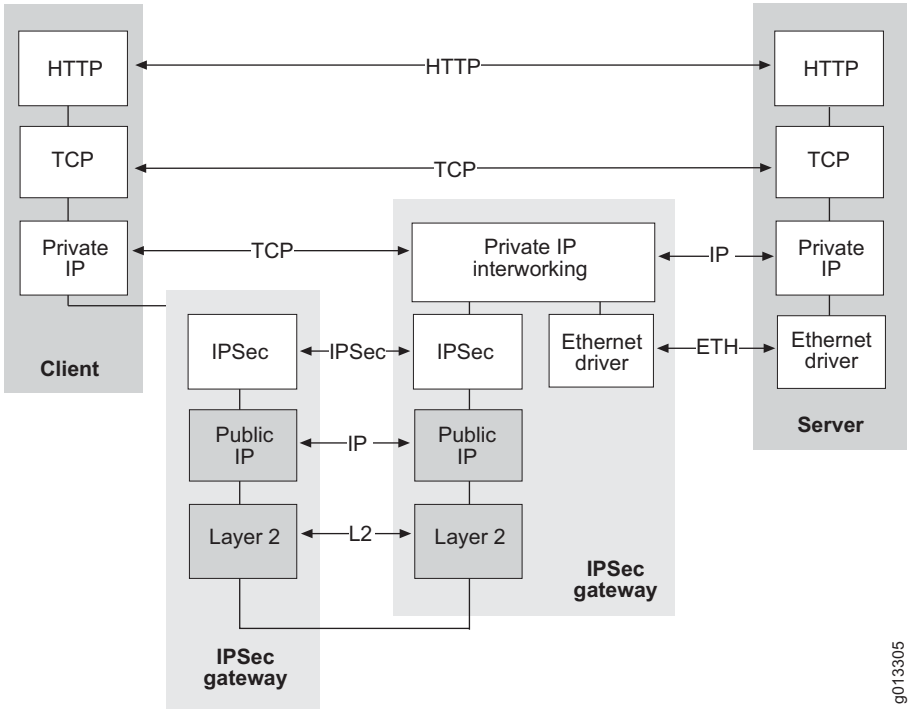
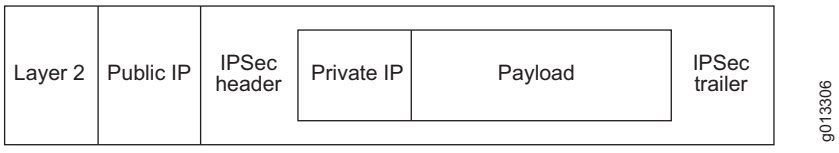


Figure 13 shows the packet encapsulation for IPSec tunneling.

Figure 13: IPSec Tunneling Packet Encapsulation



Security Parameters

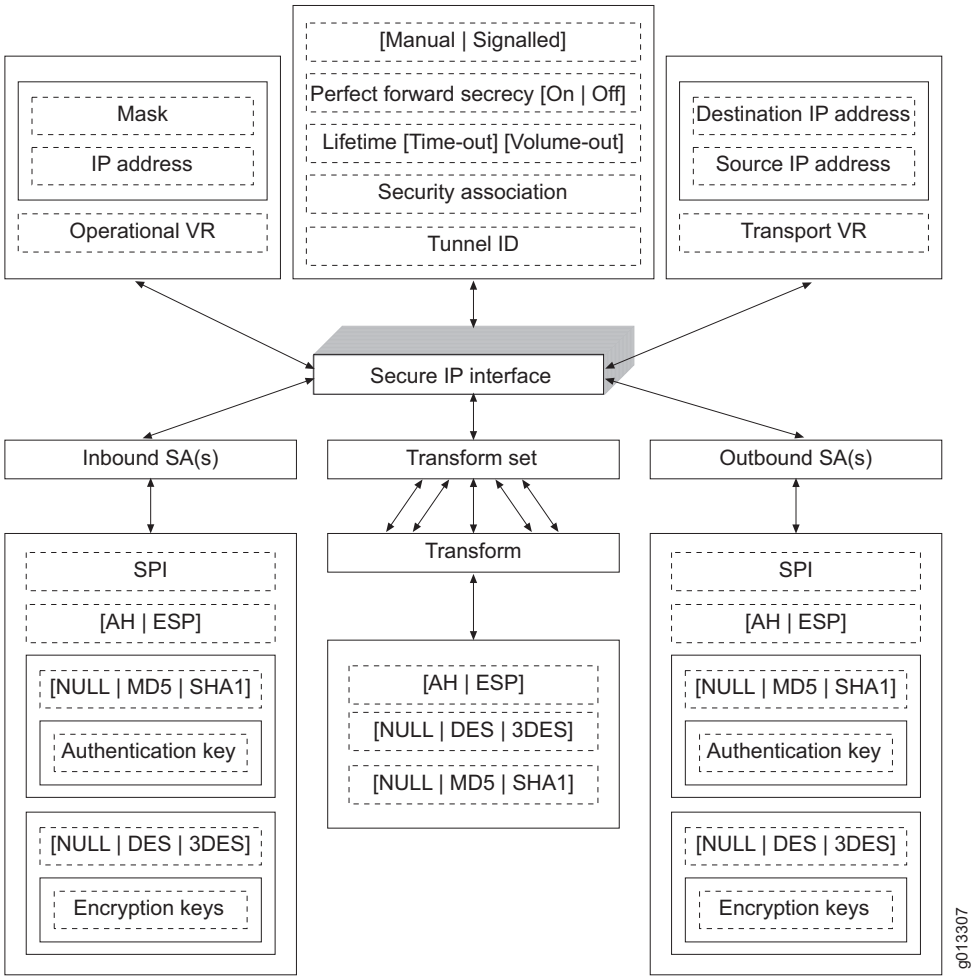
Secure IP interfaces allow tunneled traffic to be secured in many ways. For that, secure interfaces are associated with security parameters that are enforced for traffic that goes through these interfaces. Table 10 briefly describes all the parameters used for a secure IP interface.

Table 10: Security Parameters Used on Secure IP Interfaces

Security Parameter	Description
Manual or signaled	A secure IP interface, which can be either manual or signaled. ■ You can configure manual interfaces manually on both local and remote security gateways. ■ Signaled interfaces can dynamically set up connections between security gateways using ISAKMP/IKE.
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 (PFS)	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 does 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. You can manually define the SA for manual secure IP tunnels or the SA can dynamically negotiate 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 14 on page 147 shows the relationships of the various security parameters to the IPSec security interface. The following sections discuss each parameter in detail.

Figure 14: IPSec Security Parameters in Relation to the Secure IP Interface



Manual Versus Signaled Interfaces

The router 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 do not use a preconfigured, manual secure interface, the interface drops all traffic it receives. The router keeps statistics for dropped traffic. Both peer security gateways must contain a manually provisioned manual secure IP tunnel.

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

  The router 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 11 presents how the other security parameters fit with manual and signaled interfaces.

Table 11: Security Parameters per IPSec Policy Type

Security Parameter	Manual	Signaled
Operational VR	Required	Required
Transport 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 on which 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 module (ISM) 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. For IKE signaled IPSec tunnels, you can use the fully qualified domain name (FQDN) instead of the IP address to identify the tunnel endpoints. You typically use this feature to identify the tunnel destination endpoint in DSL and broadband environments. See *Transport VR Definitions with an FQDN* in this section.

- The tunnel source IP address must be one of the local IP addresses configured on the router.
- 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—Name of the transport virtual router. If not explicitly configured, the operational VR is assumed.
- Tunnel source endpoint—IP address or FQDN used as the tunnel source endpoint on this end of the tunnel. In the case of signaled tunnels, the router monitors and transmits on port 500 of this address for IKE negotiations. The tunnel source endpoint must be a configured IP address or FQDN on the transport VR, or the router indicates an error. See *Transport VR Definitions with an FQDN* for information about using an FQDN rather than an IP address.
- Tunnel destination endpoint—IP address or FQDN 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.

Transport VR Definitions with an FQDN

For signaled IPSec tunnels, you can use an FQDN instead of the IP address to specify tunnel endpoints. You typically use this feature to identify the tunnel destination in broadband and DSL environments in which the destination does not have a fixed IP address. The remote device uses the FQDN to establish and authenticate the IPSec connection, and then uses the actual IP address for rekeying and filtering operations.

The ERX router FQDN feature supports both preshared keys and digital certificates. If it uses preshared keys, the router must use IKE aggressive mode to support FQDNs.

An identity string can include an optional *user@* specification that precedes the FQDN. The entire string can be a maximum of 80 characters. For example, both of the following are supported:

```
branch245.customer77.isp.net
user4919@branch245.customer77.isp.net
```

With preshared key authentication, and when using the *user@fqdn* format, the router searches for the key based on the entire identity string. If the router cannot find that string, the router strips off the *user@* part and performs a second search based on the FQDN part of the string.

With digital certificates, the two sides of the tunnel must use the same identity format, with or without the *user@* specification; no stripping operation and no second search occurs.

NOTE: The E-series router does not support FQDN-to-IP address resolution by DNS.

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 router mandates PFS during SA negotiation. The remote security gateway must accept PFS to successfully negotiate the SA. However, if PFS is disabled, PFS might 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, 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 about setting the IKE SA lifetime, see *Lifetime* on page 160.

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 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 router finishes processing that packet.
- The actual user SA lifetime may not equal the value configured in the router.
- There are both global and tunnel-specific lifetime parameters. If there is no tunnel-specific lifetime configured, the router 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. How SA parameters are set depends on how the IP interfaces are secured:

- 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 even granted read access.

Similarly to IPSec SAs, 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—The SPI is a unique identifier that is applied to the SA when securing a 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 the encapsulation protocol. See *Transform Sets* for more information.
- Keys—The session key is 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 12.

- 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. 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. Also, tunnel mode allows an IP interface to be created and stacked right above it.

Transport mode does not add a second IP header and does not allow an IP interface to be created and stacked right above it. Instead, transport mode allows other tunneling applications, such as an L2TP tunnel, to be created and stacked on top of an IPSec transport mode connection. See *Chapter 13, Securing L2TP and IP Tunnels with IPSec* for a description of L2TP transport mode.

Supported Transforms

Table 12 describes the supported transforms.

Table 12: 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 13 lists the security functions achieved with the supported transforms, and provides a view of which combinations can be used, depending on security requirements.

Table 13: Supported Security Transform Combinations

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

The ISM 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 router uses transform number one first. If the remote system does not agree on the transform, the router 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 router compares the proposed transform from the remote end against each transform in the transform set. If there is no match, the router 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

The following sections briefly describe other supported security features for the ERX routers. These features include the following:

- IP Security Policies
- ESP Processing
- AH Processing

This section also provides a pointer to the IPSec system maximums.

IP Security Policies

The ERX router does not support a systemwide SPD. Instead, the router takes advantage of routing to forward traffic to and from a secure tunnel. The router still applies IPSec selectors to traffic going into or coming out of a secure tunnel so that unwanted traffic is not allowed inside the tunnel. Supported selectors include IP addresses, subnets, and IP address ranges.

ESP Processing

The router supports both the encryption and authentication functions of ESP encapsulation as defined in RFC 2406. Specifically, the router 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 router supports AH encapsulation as defined in RFC 2402. Specifically, the router supports:

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

IPSec Maximums Supported

See *JUNOSe Release Notes*, *Appendix A*, *System Maximums* corresponding to your software release for information about maximum values.

DPD and IPSec Tunnel Failover

Dead peer detection (DPD) is a keepalive mechanism that enables the E-series router to detect when the connection between the router and a remote IPSec peer has been lost. DPD enables the router to reclaim resources and to optionally redirect traffic to an alternate failover destination. If DPD is not enabled, the traffic continues to be sent to the unavailable destination.

When a disconnected state is detected between the E-series router and an IPSec peer, the router:

- Tears down the IPSec connection and displays the interface's state as down in output for the **show ipsec tunnel detail** command
- Clears all SAs that were established between the two endpoints
- Stops forwarding packets to the unavailable destination
- Generates SNMP traps
- Allows routing protocols running on the IP interfaces on top of the failed IPSec tunnel to switch to alternate paths
- (Optional) Redirects traffic to an alternate tunnel destination

Unlike other keepalive and heartbeat schemes, which require that peers frequently exchange Hello packets with each other at regular predetermined intervals, DPD uses two techniques to verify connectivity on an as-needed basis. In the first method, the router sends DPD inquiries to the remote peer when traffic has been sent to the peer in the last 30 seconds but no traffic has been received from the peer in the last 60 seconds. In the second method, DPD uses an idle timer. If there has been no traffic between the router and the peer for 2.5 minutes, DPD sends an inquiry to the remote end to verify that the peer is still reachable.

NOTE: Not all IPSec connections need to verify connectivity between peers. For example, the ERX router does not use DPD to check secure remote access connections based on L2TP over IPSec, which have their own keepalive mechanism. However, the router does reply to a request from a remote peer in this type of connection.

Tunnel Failover

The ERX router provides a failover mechanism for IPSec tunnels that works in concert with both DPD and with IKE SA negotiation. The tunnel failover feature provides an alternate tunnel destination when DPD detects that the current destination is unreachable or when IKE SA set up is unsuccessful. During failover, the IPSec tunnel switches to the alternate destination and establishes IPSec SAs with the new peer. To configure tunnel failover, you specify the tunnel destination backup endpoint.

Tunnel failover is a two-way process. If the router detects that the remote peer is unreachable, it switches to sending traffic to the backup destination. Likewise, if the router is sending traffic to the backup destination when the connection is terminated, the router switches to sending the traffic to the original remote peer.

NOTE: Even without tunnel failover configured, DPD still provides many benefits, such as indicating that the destination interface is down, ensuring that the router stops sending packets to the unreachable destination, and generating SNMP traps.

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 router.

Main Mode and Aggressive Mode

IKE phase 1 negotiations are used to establish IKE SAs. These SAs protect the IKE phase 2 negotiations. IKE uses one of two modes for phase 1 negotiations: main mode or aggressive mode. 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.
  - Enables greater proposal flexibility than aggressive mode.
  - Is more time consuming than aggressive mode because more messages are exchanged between peers. (Six messages are exchanged in main mode.)
- Aggressive mode
  - Exposes identities of the peers to eavesdropping, making it less secure than main mode.
  - Is faster than main mode because fewer messages are exchanged between peers. (Three messages are exchanged in aggressive mode.)
  - Enables support for fully qualified domain names (FQDNs) when the router uses preshared keys.

The next section describes aggressive mode in more detail.

Aggressive Mode Negotiations

During aggressive mode phase 1 negotiations, the E-series router behaves as follows:

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

Table 14 outlines the possible combinations of initiator proposals and policy rules. As indicated, allowing aggressive mode in a policy rule allows negotiation to take place no matter what the initiator requests.

Table 14: Initiator Proposals and Policy Rules

Aggressive Mode Setting	Initiator Requests (First Time)	Initiator Requests (Rekeyed)	Responder Policy Rule
Accepted	Main mode	Follows First Time	Aggressive or Main modes (follows initiator)
Requested	Aggressive mode	Follows First Time	Aggressive or Main modes (follows initiator)
Required	Aggressive mode	Aggressive Mode	Aggressive mode
None	Main mode	Main Mode	Main mode

The router 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 are not able to successfully negotiate the IKE SA, and no data flow is possible.

IKE policies are global to the router. Every ISM on a router 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 router uses is always the same and independent of which peer the router is negotiating with.

During negotiation, the router might 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. However, 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 applied 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 the 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. The ERX router supports two authentication methods:

- Digital certificates (using RSA algorithms)

  For digital certificate authentication, an initiator signs message interchange data using his private key, and a responder uses the initiator's public key to verify the signature. Typically, the public key is exchanged via messages containing an X.509v3 certificate. This certificate provides a level of assurance that a peer's identity (as represented in the certificate) is associated with a particular public key.

  For more information, see *Chapter 9, Configuring Digital Certificates*.

- Preshared keys

  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 router allows preshared keys to be up to 256 ASCII alphanumeric characters.

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 does not last indefinitely. Therefore, the router 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.

IKE SA Negotiation

As the initiator of an IKE SA, the router 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 router receives all IKE policies from a remote security gateway. The router then scans its own list of IKE policies to determine 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.

Generating Private and Public Key Pairs

When any of the public key methods for authenticating remote security gateways is used, the system must have at least one valid pair of public or private keys. Therefore, the system provides a facility by which it can generate public and private key pairs for itself.

The private key is used only by the system itself. It is never exchanged with any other nodes. When generated, the private key is securely stored internally to the system in nonvolatile memory. Access to the private key is never given, not even to a system administrator or to a network management system.

The public key is used in either of the following scenarios:

- A network administration system or system administrator can retrieve it so that it can be entered into remote security gateways with which the system needs to establish an IKE SA.
- It can be given to CAs so that they can properly sign it. From there, the public key is distributed to remote security gateways that can handle a PKI.

The public/private key pair as provided by the system supports the RSA standard (512, 1024, or 2048 bits).

The public/private key pair is a global system attribute, regardless of how many ISMs exist in the system. Only one set of keys is available at any given time.

Configuration Tasks

This section explains the steps to configure an IPSec license and IPSec parameters, create an IPSec tunnel, and define an ISAKMP/IKE policy. The next section contains configuration examples.

Configuring an IPSec License

By default, and with no IPSec tunnel license, you can configure up to 10 IPSec tunnels on an ERX router. However, you can purchase licenses that support the following IPSec tunnel maximums:

- 1000
- 2000
- 4000
- 8000
- 16000
- 32000

The number of additional tunnels is independent of the number of ISMs installed in the router. However, the router chassis enforces the following tunnel limits:

- SRP 10G – 10,000
- SRP 40G – 20,000

license ipsec-tunnels

- Use to specify an IPSec tunnel license.

NOTE: Acquire the license from Juniper Networks Customer Services and Support or from your Juniper Networks sales representative.

- Example

  host1(config)#**license ipsec-tunnels** *license string*

- Use the **no** version to disable the license.

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 routers use to authenticate each other.

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

   After you enter a preshared key, the router encrypts the key and displays it in masked form to increase the security of the key. If you need to reenter the key, you can enter it in its masked form using this command.

   To see the masked form of the key:

   host1#**show config**
   !
   ipsec key manual pre-share 10.10.1.1
    masked-key “AAAAGAAAAAcAAAACfd+SAsaVQ6Qeopt2rJOP6LDg+0hX5cMO”
   !
   To enter the masked key:

   host1(config-manual-key)#**masked-key**
   AAAAGAAAAAcAAAACfd+SAsaVQ6Qeopt2rJOP6LDg+0hX5cMO

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 router.

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

ipsec key manual pre-share

- Use to specify that a peer use a preshared key for authentication during the tunnel establishment phase, and to display the prompt that lets you enter the preshared key. To enter a key, use the **key** command.
- Specify the peer by using its IP address or fully qualified domain name (FQDN).
  - FQDNs are supported only for signaled tunnels.
  - The router must be in aggressive mode to use FQDNs with preshared keys.
  - The identity string can include an optional *user@* specification preceding the FQDN.
- You must enter this command in the virtual router context where the IP address of the peer is defined.

- Example 1—using an IP Address

  host1(config)#**ipsec key manual pre-share ip address 10.10.1.1**
  host1(config-manual-key)#

- Example 2—using an FQDN

  host1(config)#**ipsec key manual pre-share identity branch245.customer77.isp.net**
  host1(config-manual-key)#

- Example 3—using an FQDN with *user@* specification

  host1(config)#**ipsec key manual pre-share identity user4919@branch245.customer77.isp.net**
  host1(config-manual-key)#

- Use the **no** version to delete a manually configured key from the router.

ipsec lifetime

- Use to set the global (default) lifetime in seconds or volume of traffic in kilobytes. 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.
- To display the names of the transforms that you can use in a transform set, issue the **ipsec transform-set** ***transformSetName*** **?** command.
- 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.
- Preshared keys can have up to 256 ASCII alphanumeric characters. To include spaces in the key, enclose the key in quotation marks.
- Example 1

  host1(config-manual-key)#**key dj5fe23owi8er49fdsa**

- Example 2

  host1(config-manual-key)#**key “my key with spaces”**

- There is no **no** version. To delete a key, use the **no** version of the **ipsec key manual** command.

masked-key

- Use to enter the preshared key in masked form.
- For security purposes, the router displays the key only in masked form. If you delete the key or reboot the router to factory defaults, you can use this command to reenter the key in its masked form so that the key is not visible while you enter it.
- To see the masked key, use the **show config** command.
- Example

```
host1#show config
!
ipsec key manual pre-share 10.10.1.1
 masked-key "AAAAGAAAAAcAAAACfd+SAsaVQ6Qeopt2rJOP6LDg+0hX5cMO"
!
host1#configure terminal
host1(config)#ipsec key manual pre-share 10.10.1.1
host1(config-manual-key)#masked-key
AAAAGAAAAAcAAAACfd+SAsaVQ6Qeopt2rJOP6LDg+0hX5cMO
```

- 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. Specify the address or identity of the tunnel destination endpoint.

   host1:vrA(config-if)#**tunnel destination identity branch245.customer77.isp.net**
   host1:vrA(config-if)#**exit**

NOTE: FQDNs are used when tunnel destination endpoints do not have a fixed address, as in cable and DSL environments.

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 isakmp**

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.
- Use the **transport-virtual-router** keyword to establish the tunnel on a virtual router other than the current virtual router context.
- 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 or identity of the remote tunnel endpoint.
  - For signaled IPSec tunnels in cable or DSL environments, use the FQDN to identify the remote tunnel endpoint, which does not have a fixed IP address.
  - The identity string can include an optional *user@* specification preceding the FQDN.
- Example 1

  host1(config-if)#**tunnel destination 10.10.11.12**

- Example 2

  host1(config-if)#**tunnel destination identity branch245.customer77.isp.net**

- Example 3

  host1(config-if)#**tunnel destination identity user4919@branch245.customer77.isp.net**

- 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 number of seconds, use the **seconds** keyword to specify the lifetime in the range 1800–864000. The default value is 28800 seconds.
- To configure the lifetime in amount of traffic, use the **kilobytes** keyword to specify the lifetime in the range 102400–4294967295. The default is an unlimited volume.
- 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.
- Example

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

- Use the **no** version to restore the default lifetime (28800 seconds) and an unlimited volume.

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 restore 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 restore the default MTU (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 (PFS) 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 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.
- Use the online Help to see a list of available algorithm sets.
- Each key is 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 xj82kl40WaQp03i7 5bv0k4hm23z6uPn9**

- Use the **no** version to remove inbound session keys from a tunnel.

tunnel session-key-outbound

- Use to manually configure the authentication 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.
- Use the online Help to see a list of available algorithm sets.
- The SPI is a number in the range 256–4294967295 that identifies an SA.

- Each key is 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**—Specifies to use 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 value, **isakmp**.

tunnel source

- Use to specify an existing interface address that serves as the tunnel's source address.
- For signaled IPSec tunnels in cable or DSL environments, you can optionally use an FQDN to identify the tunnel endpoint.
- 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.

Configuring DPD and IPSec Tunnel Failover

You can use the **ipsec option dpd** command to enable dead peer detection (DPD) on the router. DPD is also known as IKE keepalive. If an IPSec tunnel destination backup is configured, the router redirects traffic to the alternate destination when DPD detects a disconnection between the E-series router and the regular tunnel destination. See the **tunnel destination backup** command.

To enable DPD and create an alternate IPSec tunnel destination for failover:

1. Enable DPD on the router.

   host1(config)#**ipsec option dpd**

2. Enter virtual router mode. Specify the VR that contains the source and destination addresses assigned to the tunnel interface (that is, the transport virtual router context).

   host1(config)#**virtual-router vrA**
   host1:vrA(config)#

3. Create an IPSec tunnel, and specify the transport VR.

   host1:vrA(config)#**interface tunnel ipsec:Aottawa2boston transport-virtual-router default**
   host1:vrA(config-if)#

4. Specify the address or identity of the tunnel destination backup endpoint.

   host1:vrA(config-if)#**tunnel destination backup identity branch500.customer77.isp.net**

ipsec option dpd

- Use to enable dead peer detection (DPD) on the router. DPD is also known as IKE keepalive.
- You configure DPD on a per-virtual router basis.
- Both peers must support DPD.
- Example

  host1(config)#**ipsec option dpd**

- Use the **no** version to restore the default, which disables DPD.

tunnel destination backup

- Use to specify the address or identity of the remote IPSec tunnel endpoint that is a backup tunnel destination. When DPD detects a disconnection between the E-series router and the regular IPSec tunnel destination, the router redirects traffic to the tunnel destination backup, and vice versa.
- You can use either the IP address or fully qualified domain name (FQDN) to identify the backup IPSec tunnel, however you must use the same type of identity that is used to specify the regular tunnel destination.
  - For signaled IPSec tunnels in cable or DSL environments, use the FQDN to identify the tunnel destination backup, which does not have a fixed IP address.
  - The identity string can include an optional *user@* specification preceding the FQDN (this is also known as a user FQDN).

NOTE: If you use a FQDN to specify the IPSec tunnel destination backup, the tunnel is not initiated by the ERX router. However, the router does respond to negotiations for this backup tunnel.

- Examples

  host1(config-if)#**tunnel destination backup 10.10.11.15**

  host1(config-if)#**tunnel destination backup identity branch245.customer88.isp.net**

  host1(config-if)#**tunnel destination backup identity user4925@branch245.customer88.isp.net**

- Use the **no** version to restore the default in which the regular tunnel destination is also the backup tunnel destination.

Defining an IKE Policy

IKE policies define parameters that the router uses during IKE phase 1 negotiation.

To create an IKE policy:

host1(config)#**ipsec ike-policy-rule 3**
host1(config-ike-policy)#

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

- Allow aggressive mode negotiation.

  host1(config-ike-policy)#**aggressive-mode**

- Specify the authentication method.

  host1(config-ike-policy)#**authentication pre-share**

- Specify the encryption algorithm.

  host1(config-ike-policy)#**encryption 3des**

- Assign a Diffie-Hellman group.

  host1(config-ike-policy)#**group 5**

- Set the hash algorithm.

  host1(config-ike-policy)#**hash md5**

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

  host1(config-ike-policy)#**lifetime 360**

aggressive-mode

- Use to enable aggressive mode negotiation for the tunnel.
- If you specify aggressive mode negotiation, the tunnel proposes aggressive mode to the peer in connections that the policy initiates.
- If the peer initiates a negotiation, the tunnel accepts the negotiation if the mode matches this policy.
- Use the **accepted** keyword to accept aggressive mode when proposed by peers
- Use the **requested** keyword to request aggressive mode when negotiating with peers
- Use the **required** keyword to only request and accept aggressive mode when negotiating with peers.
- Example

  host1(config-ike-policy)#**aggressive-mode accepted**

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

authentication

- Use to specify the authentication method the router uses in the IKE policy: preshared keys or RSA signature.
- Example

  host1(config-ike-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-ike-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-ike-policy)#**group 5**

- Use the **no** version to restore the default.

hash

- Use to set the hash algorithm for the IKE policy:
  - **md5**—MD5 (HMAC variant)
  - **sha**—SHA-1 (HMAC variant)
- Example

  host1(config-ike-policy)#**hash md5**

- Use the **no** version to restore the default, **sha**.

ipsec ike-policy-rule
ipsec isakmp-policy-rule

NOTE: The **ipsec ike-policy-rule** command replaces the **ipsec isakmp-policy-rule** command, which may be removed completely in a future release.

- Use to define an 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 in the range 1–10000, with 1 having the highest priority.
- You can add up to 10 IKE policies per router.
- Example

  host1(config)#**ipsec ike-policy-rule 3**
  host1(config-ike-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 60–86400 seconds.

  host1(config-ike-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.

Enabling Notification of Invalid Cookies

The IKE protocol enables peers to exchange informational messages. The payload of these messages can be a notify type or a delete type. These messages are expected to be protected (encrypted) by the keys negotiated by the peers when they establish a security association as a result of the IKE phase 1 exchange.

If a responder peer does not recognize the initiator-responder cookie pair, it can send an invalid cookie notification message to the initiator. The responder might fail to recognize the cookie pair because it has lost the cookie, or because it deleted the cookie and then the peer lost the delete notification. Upon receipt of the invalid cookie notification, the initiator peer can delete the phase 1 state.

The ability to send the invalid cookie message is disabled by default. You can issue the **ipsec option tx-invalid-cookie** command to enable the feature on a per-transport-VR basis.

Even when you configure this feature, the E-series router does not respond when it receives an invalid cookie notification. These notifications are unprotected by a phase 1 key exchange and therefore are subject to denial-of-service (DOS) attacks. Instead, the E-series router can determine when a phase 1 relationship has gone stale by timeouts or use of dead peer detection (DPD). For this reason, this feature is useful only when the E-series router is a responding peer for non–E-series devices that cannot detect when the phase 1 relationship goes stale.

ipsec option tx-invalid-cookie

- Use to enable the router to send an invalid cookie notification to an IKE peer when the router does not recognize the initiator-responder cookie pair.
- Example

  host1(config)#**ipsec option tx-invalid-cookie**

- Use the **no** version to restore the default, disabling the ability to send an invalid cookie notification.

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.

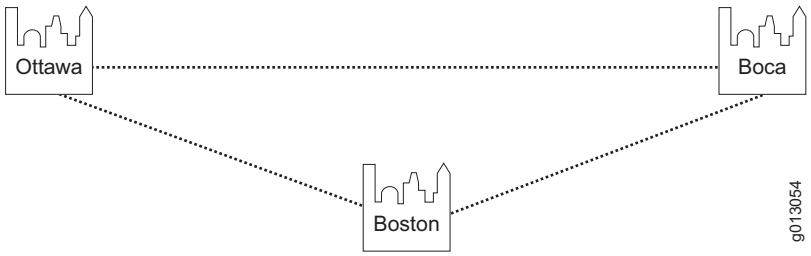
Configuration Notes

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.

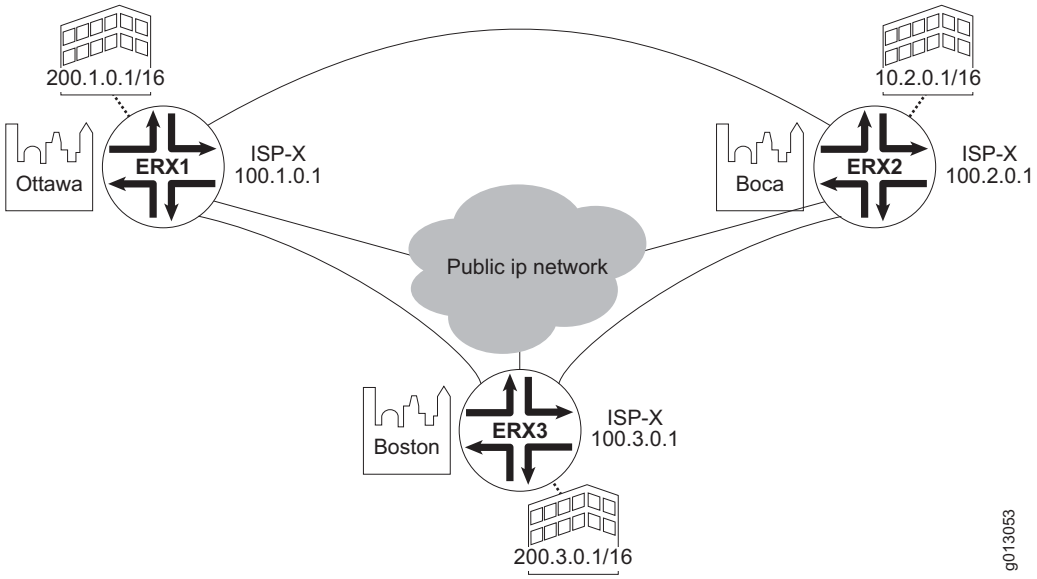
Example 1 In Figure 15 customer A is using Frame Relay to connect its corporate offices in three cities: Boston, Ottawa, and Boca.

Figure 15: 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 costs only a fraction of the price of the Frame Relay links. Figure 16 shows the connectivity scheme.

Figure 16: ISP-X Uses ERX Routers to Connect Corporate Offices over the Internet



To configure the connections as shown in Figure 16:

1. On each ERX router, 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 router, create preshared keys for the three routers to 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:

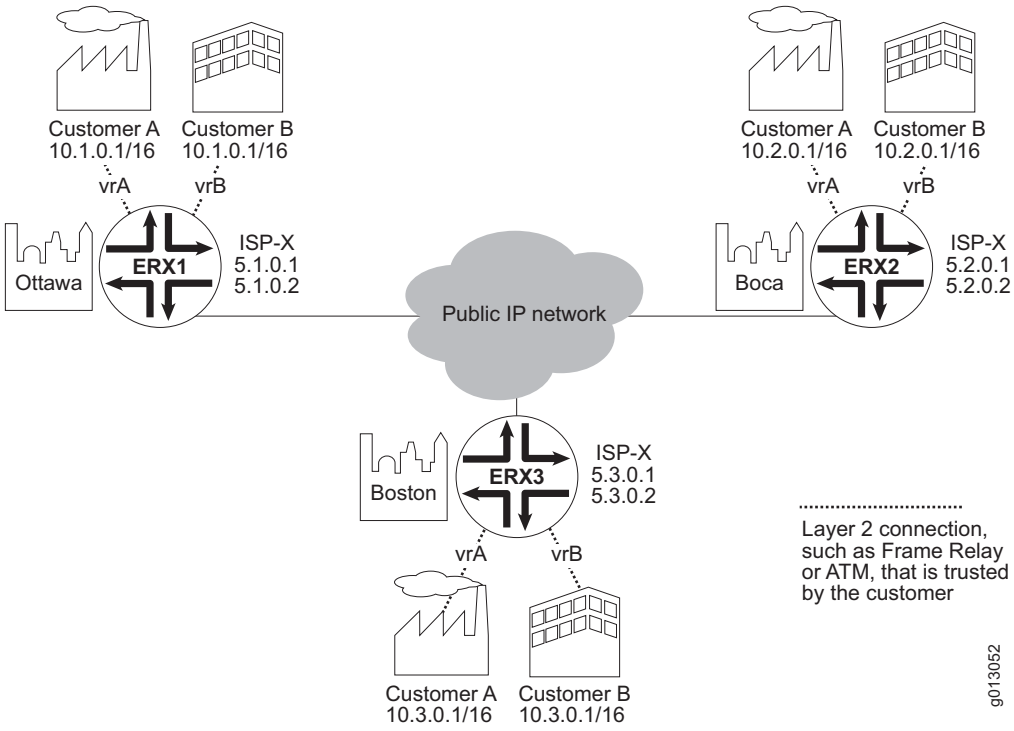
   ```
   erx3(config)#interface tunnel ipsec:Aboston2boca
   erx3(config-if)#tunnel transform-set customerAprotection
   erx3(config-if)#tunnel local-identity subnet 200.3.0.0 255.255.0.0
   erx3(config-if)#tunnel peer-identity subnet 200.2.0.0 255.255.0.0
   erx3(config-if)#tunnel source 100.3.0.1
   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 17 on page 180, 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 17: Connecting Customers Who Use Similar Address Schemes



To configure the connections as shown in Figure 17:

1. On each ERX router, 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 router, create a protection suite for the three routers to 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. Create the endpoints for the tunnels in the ISP default virtual router.

   Virtual router A:

   ```
   erx1(config)#virtual-router vrA
   erx1:vrA(config)#
   ```

   Tunnel from Ottawa to Boston on virtual router A:

   ```
   erx1:vrA(config)#interface tunnel ipsec:Aottawa2boston transport-virtual-router
   default
   erx1:vrA(config-if)#tunnel transform-set customerAprotection
   erx1:vrA(config-if)#tunnel local-identity subnet 10.1.0.0 255.255.0.0
   erx1:vrA(config-if)#tunnel peer-identity subnet 10.3.0.0 255.255.0.0
   erx1:vrA(config-if)#tunnel source 5.1.0.1
   erx1:vrA(config-if)#tunnel destination 5.3.0.1
   erx1:vrA(config-if)#ip address 10.3.0.0 255.255.0.0
   erx1:vrA(config-if)#exit
   ```

   Tunnel from Ottawa to Boca on virtual router A:

   ```
   erx1:vrA(config)#interface tunnel ipsec:Aottawa2boca transport-virtual-router
   default
   erx1:vrA(config-if)#tunnel transform-set customerAprotection
   erx1:vrA(config-if)#tunnel local-identity subnet 10.1.0.0 255.255.0.0
   erx1:vrA(config-if)#tunnel peer-identity subnet 10.2.0.0 255.255.0.0
   erx1:vrA(config-if)#tunnel source 5.1.0.1
   ```

```
erx1:vrA(config-if)#tunnel destination 5.2.0.1
erx1:vrA(config-if)#ip address 10.2.0.0 255.255.0.0
erx1:vrA(config-if)#exit
```

Virtual router B:

```
erx1(config)#virtual-router vrB
erx1:vrB(config)#
```

Tunnel from Ottawa to Boston on virtual router B:

```
erx1:vrB(config)#interface tunnel ipsec:Bottawa2boston 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 10.3.0.0 255.255.0.0
erx1:vrB(config-if)#tunnel source 5.1.0.2
erx1:vrB(config-if)#tunnel destination 5.3.0.2
erx1:vrB(config-if)#ip address 10.3.0.0 255.255.0.0
erx1: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. Create the endpoints for the tunnels 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 contains information about troubleshooting and monitoring IPSec.

System Event Logs

To troubleshoot and monitor IPSec, use the following system event logs:

- auditIpsec—Lower layers of IKE SA negotiations
- ikepki—Upper layers of IKE SA negotiations
- stTunnel—Secure tunnel interface

For more information about using event logs, see the *JUNOSe System Event Logging Reference Guide*.

show Commands

To view your IPSec configuration and to monitor IPSec tunnels and statistics, use the following **show** commands.

show ipsec ike-policy-rule
show ike policy-rule

NOTE: The **show ipsec ike-policy-rule** command replaces the **show ipsec isakmp-policy-rule** command, which may be removed completely in a future release.

- 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, 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 ipsec 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:RSA Signatures
    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 ipsec ike-sa
show ike sa

NOTE: The **show ipsec ike-sa** command replaces the **show ike sa** command, which may be removed completely in a future release.

- Use to display IKE phase 1 SAs running on the router.
- Field descriptions
  - Local:Port—Local IP address and UDP port number of phase 1 negotiation
  - Remote:Port—Remote IP address and UDP port number of phase 1 negotiation
  - Time(Sec)—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. Possible states are:
    - AM_SA_I—Initiator has sent initial aggressive mode SA payload and key exchange to the responder
    - AM_SA_R—Responder has sent aggressive mode SA payload and key exchange to the initiator
    - AM_FINAL_I—Initiator has finished aggressive mode negotiation
    - AM_DONE_R—Responder has finished aggressive mode negotiation
    - MM_SA_I—Initiator has sent initial main mode SA payload to the responder
    - MM_SA_R—Responder has sent a response to the initial main mode SA
    - MM_KE_I—Initiator has sent initial main mode key exchange to the responder

- MM_KE_R—Responder has sent a response to the key exchange
- MM_FINAL_I—Initiator has sent the final packet in the main mode negotiation
- MM_FINAL_R—Responder has finished main mode negotiation
- MM_DONE_I—Initiator has finished main mode negotiation
- DONE—Phase 1 SA negotiation is complete, as evidenced by receipt of some phase 2 messages

- Local Cookie—Unique identifier (SPI) for the local phase 1 IKE SA
- Remote Cookie—Unique identifier (SPI) for the remote phase 1 IKE SA

- Example

```
host1#show ipsec ike-sa

IKE Phase 1 SA's:
Local:Port            Remote:Port           Time(Sec) State       Local Cookie       Remote Cookie
195.0.0.100:500       195.0.0.200:500       1551      DONE        0x90ee723e6cb0c016 0xf7d3651e93d56431
195.0.0.100:500       195.0.0.200:500       1552      DONE        0x821bccf81dcedbb0 0x35152bdb7a9c734e
195.0.1.100:500       195.0.1.200:500       1687      DONE        0x1b4fbcebe36d1b16 0xed742166a305a6a0
195.0.1.100:500       195.0.1.200:500       1687      DONE        0xacf3acd1b3555b6a 0x0af9edbc95622869
195.0.2.100:500       195.0.2.200:500       1688      DONE        0x3153379b32d8c936 0x17f5d77f9badc3cf
195.0.2.100:500       195.0.2.200:500       1688      DONE        0x6573dcbc9bf31fae 0x7af8b4d13078b463
195.0.3.100:500       195.0.3.200:500       1685      DONE        0xdc7df648fcac375a 0x0346752d2881d5c5
195.0.3.100:500       195.0.3.200:500       1685      DONE        0xe776e9ffb6678635 0x8de857af1c681874
195.0.4.100:500       195.0.4.200:500       1690      DONE        0x16410d890500e94e 0xbd47831b55e81c27
```

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 option

- Use to display the status, enabled or disabled, of IPSec options configured on the current virtual router. Information is displayed for the following options:
  - Dead peer detection (DPD)
  - Network Address Translation Traversal (NAT-T). For information about configuring and monitoring NAT-T on L2TP/IPSec tunnels, see *Chapter 13, Securing L2TP and IP Tunnels with IPSec*.
  - Transmission of invalid cookie notification in ISAKMP messages to peers

- Example

```
host1:vrA#show ipsec option

IPsec options:
Dead Peer Detection: disabled
NAT Traversal      : enabled
TX Invalid Cookie  : disabled
```

show ipsec transform-set

- Use to display transform sets configured on the router.
- 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 remoteEndpoint—IP address of remote tunnel endpoint
    - Tunnel source—IP address or FQDN of tunnel source
    - Tunnel destination—IP address or FQDN of tunnel destination
    - Tunnel backup destination—Alternate 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 outbound spi/SA—SPI and SA in use on traffic sent to the tunnel (manual tunnels only)

- Tunnel inbound spi/SA—SPI and SA in use on traffic received from the tunnel (manual tunnels only)
- Tunnel lifetime seconds—Configured time-based lifetime in seconds
- Tunnel lifetime kilobytes—Configured traffic-based 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 Operational Attributes—Displays statistics related to the tunnel lifetime
  - inbound/outboundSpi/SA—SPI in use on traffic received from or sent to the tunnel
  - inbound/outboundSa—SA in use on traffic received from or sent to the tunnel
  - inbound/outbound lifetime allowed—Negotiated time-based lifetime in seconds
  - inbound/outbound lifetime remaining—Number of seconds remaining before time-based lifetime expires
  - inbound/outbound traffic allowed—Negotiated traffic-based lifetime in kilobytes
  - inbound/outbound traffic remaining—Number of additional kilobytes that tunnel can send or receive before traffic-based lifetime expires
- 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

```
hostl#show ipsec tunnel detail
IPSEC tunnel r200000 is Up
Tunnel configuration:
  Tunnel type is signaled
  Tunnel mtu is 1440
  Tunnel local endpoint is 195.0.0.200
  Tunnel remote endpoint is 195.0.0.100
  Tunnel source is 195.0.0.200
  Tunnel destination is 195.0.0.100
  Tunnel backup destination is 0.0.0.0
  Tunnel transport virtual router is r
  Tunnel transform set is perf
  Tunnel local identity is ipAddress: 4.0.0.100
  Tunnel peer identity is ipAddress: 3.0.0.100
  Tunnel lifetime seconds is 7200
  Tunnel lifetime kilobytes is 1024000
  Tunnel pfs is group 5
  Tunnel administrative state is Up

Tunnel Operational Attributes:
  inboundSpi  = 0x17270202, inboundSa = esp-3des-hmac-sha
  inbound lifetime: allowed 7200s, remaining 7100s
  inbound traffic: allowed 1024000KB, remaining 1023997KB

  outboundSpi = 0x283b0201,  outboundSa = esp-3des-hmac-sha
  outbound lifetime: allowed 7200s, remaining 7100s
  outbound traffic: allowed 1024000KB, remaining 1023997KB

Tunnel Statistics:
  InUserPackets           15
  InUserOctets            1920
  InAccPackets            15
  InAccOctets             2760
  InAuthErrors            0
  InReplayErrors          0
  InPolicyErrors          0
  InOtherRxErrors         0
  InDecryptErrors         0
  InPadErrors             0

  OutUserPackets          15
  OutUserOctets           1920
  OutAccPackets           15
  OutAccOctets            2760
  OutPolicyErrors         0
  OutOtherTxErrors        0
```

show ipsec tunnel summary

- Use to display a summary of all tunnels configured on the router.
- Field descriptions
  - Total number of ipsec interface—Number of tunnels configured on the router
  - 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 s0l1e3d0 is up
IPSEC tunnel s0l1e3d1 is up
IPSEC tunnel s0l2e3d0 is up
IPSEC tunnel s0l2e3d1 is up
IPSEC tunnel s0l3e3d0 is up
IPSEC tunnel s0l4e3d0 is up
IPSEC tunnel s0l4e3d1 is up
IPSEC tunnel s0l5e3d0 is up
```

show license ipsec-tunnels

- Use to display the IPSec license key configured on the router and the number of tunnels allowed on the router.
- Example

```
host1#show license ipsec-tunnels
ipsec-tunnels license is 'g1k23b23eb2j' which allows 5000 tunnels with 1
IPsec card and 7500 tunnels with 2 or more IPsec cards.
```