Configuring and Using TCP/IP Services
for OpenVMS IPsec

hp ®

i n v e n t

Table of Contents

List of Figures

List of Tables

List of Examples

About This Document

This manual explains how to set up, manage, and troubleshoot IPsec, the optional Internet Security feature that is provided with Version 5.7 of TCP/IP Services for OpenVMS.

The HP TCP/IP Services for OpenVMS product is the HP implementation of the TCP/IP networking protocol suite and internet services for OpenVMS Alpha and I64 systems.

TCP/IP Services provides a comprehensive suite of functions and applications that support industry-standard protocols for heterogeneous network communications and resource sharing.

NOTE: The manual is a work in progress, intended for Early Adopters Kit users only, and not for official release.

Intended Audience

This manual is intended for experienced OpenVMS system and network administrators responsible for installing, configuring, and managing TCP/IP Services for OpenVMS IPsec and assumes a working knowledge of OpenVMS system management, TCP/IP network, and TCP/IP terminology. It is helpful to have knowledge of Transmission Control Protocol/Internet Protocol (TCP/IP) networking concepts and network configuration.

New and Changed Information in This Edition

This is a new manual.

Document Structure

This guide describes how to configure, manage, and solve problems with the IPsec software. It contains the following chapters and appendixes:

- Chapter 1, Overview, introduces features, definitions and concepts that are important to understanding IPsec.
- Subsequent chapters TBS.

Conventions

The following conventions may be used in this manual:

Convention	Meaning
Ctrl/*x*	A sequence such as Ctrl/*x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key (*x*) or a pointing device button.
Return	In examples, a key name in bold indicates that you press that key.
…	A horizontal ellipsis in examples indicates one of the following possibilities: – Additional optional arguments in a statement have been omitted. – The preceding item or items can be repeated one or more times. – Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

Convention	Meaning
( )	In command format descriptions, parentheses indicate that you must enclose choices in parentheses if you specify more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
		In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are optional; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold type	Bold type represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic type	Italic type indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*), in command lines (/PRODUCER=*name*), and in command parameters in text (where (*dd*) represents the predefined par code for the device type).
UPPERCASE TYPE	Uppercase type indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Example`	This typeface indicates code examples, command examples, and interactive screen displays. In text, this type also identifies URLs, UNIX commands and pathnames, PC-based commands and folders, and certain elements of the C programming language.
–	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

Related Information

The following manuals describe how to install, customize, and use TCP/IP Services:

- *HP TCP/IP Services for OpenVMS Concepts and Planning*

  This manual provides conceptual information about TCP/IP networking on OpenVMS systems, including general planning issues to consider before configuring your system to use the TCP/IP Services software.

  This manual also describes the manuals in the TCP/IP Services documentation set and provides a glossary of terms and acronyms for the TCP/IP Services software product.

- *HP TCP/IP Services for OpenVMS Release Notes*

  The release notes provide version-specifiec information that supersedes the information in the documentation set. The features, restrictions, and corrections in this version of the software are described in the release notes. Always read the release notes before installing the software.

- *HP TCP/IP Services for OpenVMS Installation and Configuration*

  This manual explains how to install and configure TCP/IP Services.

- *HP TCP/IP Services for OpenVMS User's Guide*

  This manual describes how to use the applications available with TCP/IP Services such as remote file operations, email, TELNET, TN3270, and network printing.

- *HP TCP/IP Services for OpenVMS Management*

  This manual describes how to configure and manage the TCP/IP Services product.

- *HP TCP/IP Services for OpenVMS Management Command Reference*

  This manual describes TCP/IP Services management commands.

- *HP TCP/IP Services for OpenVMS Management Command Quick Reference Card*

  This reference card lists the TCP/IP management commands by component and describes the purpose of each command.

- *HP TCP/IP Services for OpenVMS UNIX Command Equivalents Card*

  This reference card contains information about commonly performed network management tasks and their corresponding TCP/IP management and UNIX command formats.

- *HP TCP/IP Services for OpenVMS ONC RPC Programming*

  This manuals presents an overview of high-level programming using open network computing remote procedure calls (ONC RPCs). This manual also describes the RPC programming interface and how to use the RCPGEN protocol compiler to create applications.

- *HP TCP/IP Services for OpenVMS Guide to SSH*

  This manual describes how to configure, set up, use, and manage the SSH for OpenVMS software.

- *HP TCP/IP Services for OpenVMS Sockets API and System Services Programming*

  This manual describes how to use the Sockets API and OpenVMS system services to develop network applications.

- *HP TCP/IP Services for OpenVMS SNMP Programming and Reference*

  This manual describes the Simple Network Management Protocol (SNMP) and the SNMP application programming environment (eSNMP). It describes the subagents provided with TCP/IP Services, utilities provided for managing subagents, and how to build your own subagents.

- *HP TCP/IP Services for OpenVMS Tuning and Troubleshooting*

  This manual provides information about how to isolate the causes of network problems and how to tune the TCP/IP Services software for the best performance.

- *HP TCP/IP Services for OpenVMS Guide to IPv6*

  This manual describes the IPv6 environment, the roles of systems in this environment, the types and function of the different IPv6 addresses, and how to configure TCP/IP Services to access the IPv6 network.

- *HP TCP/IP Services for OpenVMS Guide to IPsec*

  This manual describes how to set up, manage, and solve problems with the IPsec internet security software.

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

feedback@fc.hp.com

Include the document title, manufacturing part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 OpenVMS IPsec Overview

Features

The IP security (IPsec) protocol suite was defined by the Internet Engineering Task Force (IETF) to provide security for IP networks. OpenVMS IPsec is the HP implementation of IPsec. OpenVMS IPsec provides the following security services for IP networks:

- **Data integrity and authentication**

  The IPsec **Authentication Header** (**AH**) provides data integrity and authentication to prevent unauthorized creation, modification, or deletion of transmitted data. The AH header also includes a sequence number for replay protection. IPsec can also verify that the claimed sender is the actual sender. The AH does not provide privacy—the IP data is not encrypted.

- **Data Privacy**

  The IPsec **Encapsulating Security Payload** (**ESP**) encrypts IP data to provide data privacy. ESP also provides data authentication and integrity. The ESP header also includes a sequence number for replay protection. On gateways, IPsec can also be used to encapsulate and encrypt the original IP packet to protect the identity of the end source and destination IP addresses.

- **Application-transparent security**

  You do not need to rewrite or reconfigure applications to use OpenVMS IPsec. IPsec security headers are inserted between the standard IP protocol header and the upper-layer data (such as a TCP packet). Any network service that uses IP (such as `telnet`, `FTP`, `SMTP`, or `IGMP`) or user applications that use IP can use IPsec without modification.

  IPsec traffic can also pass transparently through existing IP routers.

- **Dynamic encryption key management**

  OpenVMS IPsec supports the **Internet Key Exchange** (**IKE**) protocol, part of the IPsec protocol suite, to establish and manage dynamic cryptographic keys. Using dynamic keys (keys that change) to encrypt and authenticate data provides additional security.

- **Identity authentication**

  The IKE protocol authenticates the identity of the remote system. IPsec supports the following forms of IKE authentication:

  — Preshared keys.
  — Digital signatures (RSA signatures), using X.509 version 3 security certificates.

  > **NOTE:**   Digital signatures are not supported in this EAK release.

  Because IKE verifies the identity of the remote system, AH and ESP provide data origin authentication.

- **Host-based IPsec topologies**

  OpenVMS IPsec is supported on host systems in host-to-host and in host-to-gateway topologies. You can use IPsec to provide security in internal networks and to provide Virtual Public Network (VPN) solutions across public Internet communication.

  You can also use OpenVMS IPsec with application servers (proxy application servers) and IPsec VPN gateways from other vendors.

- **Powerful and flexible management utilities**

  The OpenVMS IPsec product includes the configuration and management features listed below.

— **Easy-to-use configuration utilities**

You configure OpenVMS IPsec using **`ipsec_config`**, which allows batch mode operation.

— **Flexible, packet-based configuration**

You control IPsec behavior by defining packet filters in IPsec policies. An IPsec policy contains a packet filter definition and list of actions or transforms (pass, discard, use ESP or AH) to apply to the packets. The packet filter definition contains the following fields:

◦ local IP address
◦ local address prefix length (for subnet addresses)
◦ remote IP address
◦ remote address prefix length (for subnet addresses)
◦ upper-layer protocol (such as TCP, UDP. or ICMP)
◦ local TCP or UDP port number
◦ remote TCP or UDP port number

You can specify wildcards (match any value) for field values. You can also select a network service for the filter, such as `telnet`, instead of the upper-layer protocol and port numbers.

IPsec Protocol Suite

The major components of the IPsec protocol suite can be divided into the following categories:

- **Encapsulating Security Payload (ESP)** header for data confidentiality, data integrity, and data authentication. The ESP header also includes a sequence number that provides a form of replay protection.
- **Authentication Header (AH)** for data integrity and authentication. The AH header also includes a sequence number for a form of replay protection.
- **Internet Key Exchange (IKE)** protocol, for generating and distributing cryptography keys for ESP and AH. IKE also authenticates the identity of the remote system, so AH and authenticated ESP with IKE keys provides data origin authentication.
- **Manual Keys** , an alternative to IKE. Instead of dynamically generating and distributing cryptography keys for ESP and AH, the cryptography keys are static and manually distributed. Manual keys are typically used only when the remote system does not support IKE.

Encapsulating Security Payload (ESP)

The IPsec Encapsulating Security Payload (ESP) uses shared key encryption to provide data privacy and shared key hash functions to provide data authentication and data integrity.

Shared Key Encryption

In **shared key encryption** , two parties know the same cryptographic key. The sender (System A in Figure 1-1) encrypts the data with the key to create encrypted data. The recipient (System B in Figure 1-1) decrypts the encrypted data with the same key. Since only a holder of the cryptographic key can decrypt the data, the encrypted data can be transmitted across the network without being understood by other parties.

Figure 1-1 Shared Key Encryption
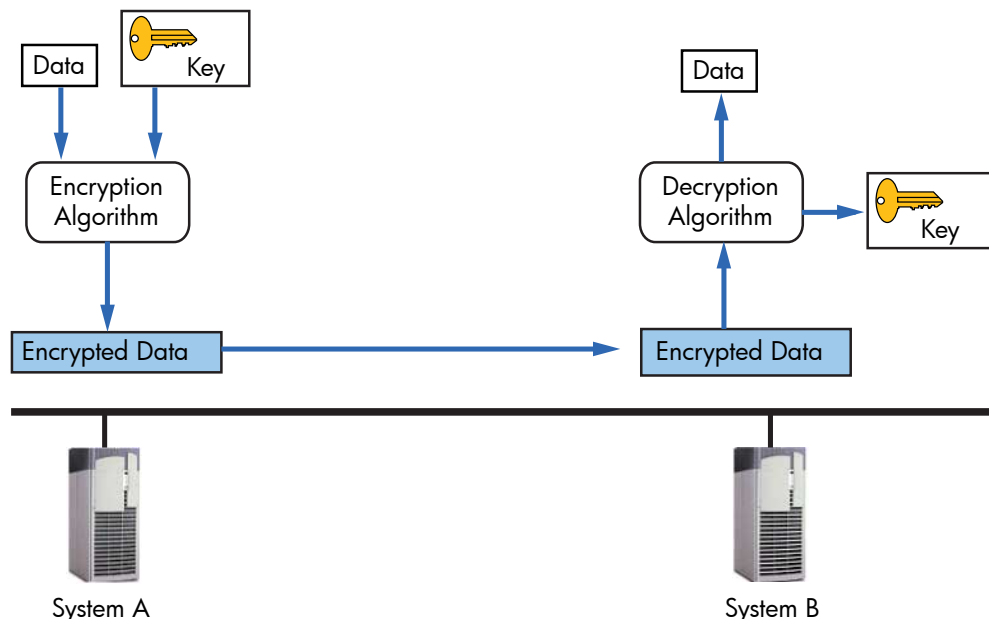
Shared key cryptography alone does not provide protection against tampering. An intruder can still intercept encrypted data and alter it before sending it to the correct destination. For this reason, ESP also authenticates the encrypted data.

Shared key cryptography is also referred to as **symmetric key cryptography** (because the keys used by both parties must be the same) and **private key cryptography** (because the two parties must keep the key private).

Shared Key Hash Functions

Shared key hash functions (also known as a symmetric key hash functions) are hash functions that take a large block of variable-length data and a shared key as input and produce a small, fixed-length hash value, or authentication code. The IPsec protocol suite uses a specific method for producing the hash value and refers to the authentication value as the **Hashed Message Authentication Code** (**HMAC**).

Shared keyed hash functions are usually based on one-way hash functions: Starting with a hash output value, it is difficult to create an input value that would generate the same output value, even if no key is used. This makes it difficult for a third party to intercept a message and replace it with a new message that generates the same authentication code. This ensures that only a holder of the secret key can generate the correct authentication code.

The sender uses the plaintext (data) and the shared key to calculate an HMAC for the data and sends the HMAC with the data. The recipient computes its own HMAC value using the same shared secret key and data. The recipient then compares the result with the transmitted HMAC. If the HMAC values match, the recipient is assured that the sender knows the same secret key, confirming the identity of the sender. The recipient is also assured that the data was not altered during transit.

ESP Processing

On the sender, the ESP module processes the outbound packet as follows:

1. The ESP module encrypts the IP payload using the encryption key.
2. The ESP module collates an authentication value (the HMAC), for the encrypted payload using the authentication key and appends the authentication value to the packet.

On the remote system, the recipient ESP module processes the inbound ESP packet as follows:

1. The recipient ESP module calculates its own authentication value for the encrypted payload using its copy of the authentication key.
2. The recipient ESP compares its authentication value with the transmitted authentication value (the HMAC). If the values match, the recipient then uses its copy of the encryption key to decrypt the encrypted portion of the packet and extract the original payload.

Transport and Tunnel Modes

ESP can be used in transport mode or tunnel mode.

Transport Mode

In transport mode, IPsec inserts the ESP header after the original IP header, and adds the ESP trailer and authentication value to the end of the packet. Only the IP payload (e.g., TCP, UDP, or IGMP packet) is secured (encrypted and authenticated). The IP header is not secured. Transport mode is typically used for end-to-end security. Figure 1-2 shows IPv4 ESP packets in transport mode.

Figure 1-2 ESP Transport Mode
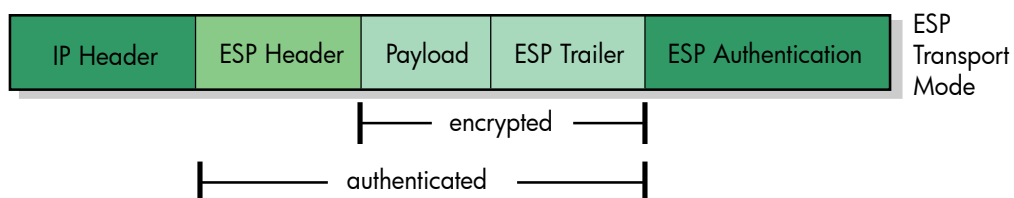
Tunnel Mode

In **tunnel mode** , IPsec encloses, or encapsulates, the original IP packet, including the original IP header, within a second IP datagram. All of the original IP packet, including the original

header, is secured. Tunnel mode is typically used on secure gateways. When ESP is used in tunnel mode on gateways, the outer, unencrypted IP header contains the IP addresses of the gateways, and the inner, encrypted IP header contains the end IP source and destination addresses. This prevents eavesdroppers from detecting or analyzing traffic between the end source and destination addresses. Figure 1-3 shows IPv4 ESP packets in tunnel mode.

Figure 1-3 ESP Tunnel Mode

New IP Header	ESP Header	IP Header	Payload	ESP Trailer	ESP Authentication	ESP Tunnel Mode

encrypted (IP Header — ESP Trailer)
authenticated (ESP Header — ESP Authentication)

IPv6 ESP Transport Mode

In IPv6 ESP transport mode (shown in Figure 1-4), IPsec inserts the ESP header after the following headers and extensions:

- the basic IPv6 header
- hop-by-hop options
- any destination options needed to interpret the ESP header
- routing extensions
- fragment extensions

The items listed below follow the ESP header and are encrypted and authenticated:

- any destination options needed only for the "final" destination and not needed to interpret the ESP header
- the IP data or payload (e.g., TCP or UDP packet)

Figure 1-4 IPv6 ESP in Transport Mode

IP Header	Extension Headers (a)	ESP Header	Destination Options (b)	Payload	ESP Trailer	ESP Authentication	ESP Transport Mode

encrypted (Destination Options — ESP Trailer)
authenticated (ESP Header — ESP Authentication)
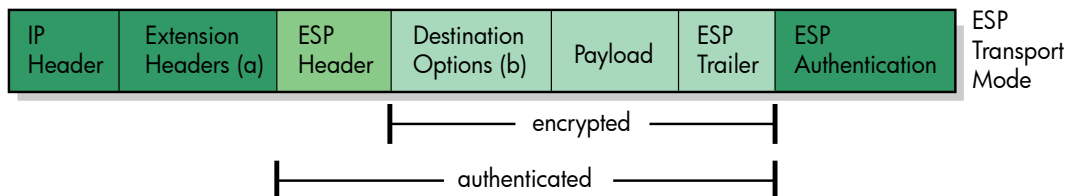
IPv6 ESP Tunnel Mode

In IPv6 ESP tunnel mode (shown in Figure 1-5), the packet layout is the same as IPv4 ESP tunnel mode, except that the original and new (outer) IP headers may include header extensions.

Figure 1-5 IPv6 ESP in Tunnel Mode

New IP Header	New Extension Headers	ESP Header	IP Header	Extension Headers	Payload	ESP Trailer	ESP Authentication	ESP Tunnel Mode

encrypted (IP Header — ESP Trailer)
authenticated (ESP Header — ESP Authentication)

ESP Encryption and Authentication Algorithms

OpenVMS IPsec supports the encryption algorithms listed in Table 1-1 (page 22) and the authentication algorithms listed in Table 1-2 (page 22). For example, OpenVMS IPsec can encrypt an ESP packet using AES and authenticate it using SHA1.

Table 1-1 OpenVMS IPsec Encryption Algorithms

Name	Description
AES	Advanced Encryption Standard (AES) Cipher Block Chaining (CBC) mode encryption using a 128-bit key.
DES	Data Encryption Standard (DES) CBC encryption using a 56-bit key.
3DES	Triple-DES CBC, three CBC encryption iterations, each with a different 56-bit key.

Table 1-2 OpenVMS IPsec Authentication Algorithms

Name	Description
MD5	Message Digest-5, 160-bit key
SHA1	Secure Hash Algorithm-1, 128-bit key

⚠️ **WARNING!** DES-CBC has been cracked (data encoded by DES has been decoded by a third party). HP recommends that you use DES only when you are required to so for compatibility reasons or because of legal restrictions.

💡 **TIP:** HP recommends that you use AES128 with SHA1. AES is the most secure form of encryption for OpenVMS IPsec, and SHA1 is considered more secure than MD5.

AES encryption throughput rates are comparable to or better than DES and 3DES rates.

Non-Authenticated ESP

ESP encryption takes the data carried by IP, such as a TCP packet, and encrypts it using a cryptographic key. The receiving IPsec ESP entity uses the same key to decrypt the cipher text and extract the original data.

Authentication Header (AH)

The IPsec Authentication Header (AH) provides integrity and authentication but no privacy—the IP data is not encrypted. The AH contains an authentication value based on a symmetric-key hash function. Because AH does not encrypt data, it is not commonly used. However, AH provides one feature that ESP does not: AH authenticates non-mutable fields in the IP header (fields that do not change in transit, including source and destination addresses). For this reason, AH is sometimes used with ESP, by nesting an ESP packet within an AH packet.

OpenVMS IPsec supports the following authentication algorithms for AH :

- **HMAC-SHA1**
- **HMAC-MD5**

Transport and Tunnel Modes

AH can be used in transport mode or tunnel mode.

Transport Mode

In transport mode, IPsec inserts the AH header with the authentication value after the IP header. The IP data and header are used to calculate the AH authentication value. Mutable fields in the IP header (fields might change in transit), such as "hop count," and "time to live," are assigned a zero value before IPsec calculates the authentication value, so the actual values of the mutable fields are not authenticated. Figure 1-6 shows AH in transport mode.

Figure 1-6 AH in Transport Mode

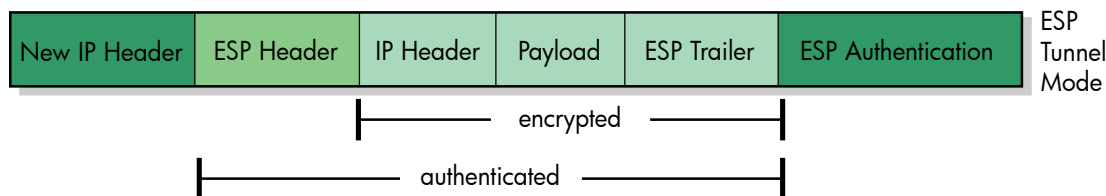| IP Header | AH Header | Payload | AH Transport Mode |
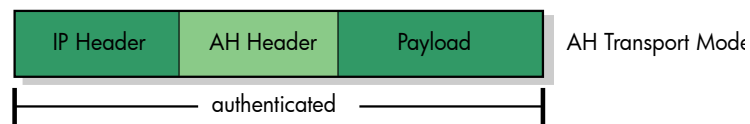
authenticated

Tunnel Mode

In tunnel mode, IPsec encloses, or encapsulates, the original IP datagram, including the original IP header, within a second IP datagram. All of the original IP datagram, including all fields of the original header, is authenticated. Figure 1-7 shows AH in tunnel mode.

Figure 1-7 AH in Tunnel Mode

| New IP Header | AH Header | IP Header | Payload | AH Tunnel Mode |

authenticated
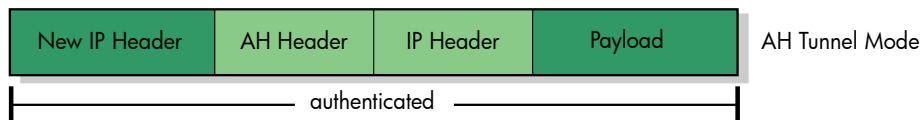
IPv6 AH Transport Mode

In IPv6 AH transport mode, IPsec inserts the AH after the following headers and extensions:

*   the basic IPv6 header
*   hop-by-hop options
*   any destination options needed to interpret the AH header
*   routing extensions
*   fragment extensions

The items listed below follow the AH:

*   any destination options needed only for the "final" destination and not needed to interpret the AH header
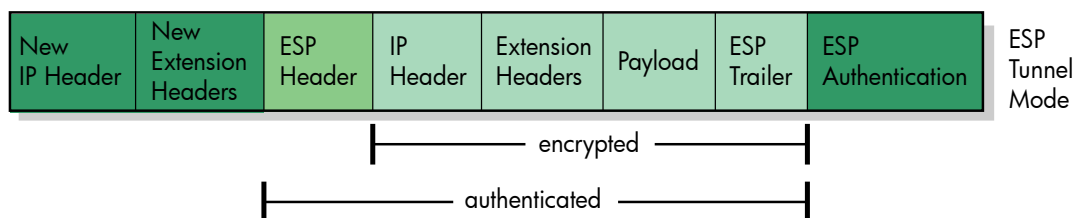*   the IP data or payload (e.g., TCP or UDP packet)

The entire packet is used to calculate the authentication value. Mutable and unpredictable fields and options, such as timestamp and traceroute options, are assigned a zero value before calculating the authentication value.

Figure 1-8 IPv6 AH Transport Mode

| IP Header | Extension Headers (a) | AH Header | Destination Options (b) | Payload | AH Transport Mode |

authenticated
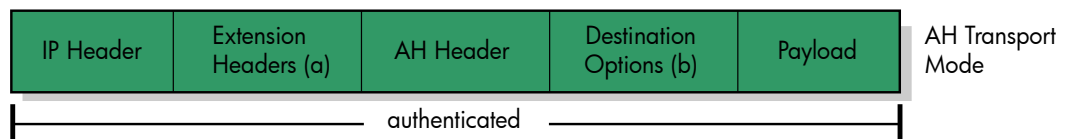
IPv6 AH Tunnel Mode

In IPv6 AH tunnel mode, the packet layout is the same as IPv4 AH tunnel mode, except that the original and new (outer) IP headers may include header extensions.

Figure 1-9 IPv6 AH Tunnel Mode

| New IP Header | New Extension Headers | AH Header | IP Header | Extension Headers | Payload | AH Tunnel Mode |
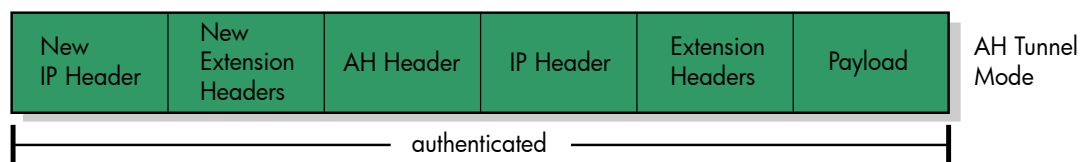
authenticated

Nested ESP in AH

An ESP packet can be nested within an AH packet. For example, an ESP packet using AES and SHA1 can be nested within an AH MD5 packet. IPsec uses a key to encrypt the payload using AES, and a second key to generate an ESP SHA1 authentication value. The ESP SHA1 authentication value authenticates the IP payload and parts of the ESP header. IPsec then nests the ESP packet within an AH packet, using a third key to generate the AH MD5 authentication value. The AH MD5 authentication value authenticate the IP packet header and payload, except the mutable fields of the IP header.

Internet Key Exchange (IKE)

Before IPsec sends authenticated or encrypted IP data, both the sender and receiver must agree on the protocols, encryption algorithms and keys to use. IPsec uses the Internet Key Exchange (IKE) protocol to negotiate the encryption and authentication methods, and generate shared encryption keys. The IKE protocol also provides **primary authentication** - verifying the identity of the remote system before negotiating the encryption algorithm and keys.

The IKE protocol is a hybrid of three other protocols: Internet Security Association and Key Management Protocol (ISAKMP), Oakley, and Versatile Secure Key Exchange Mechanism for Internet protocol (SKEME).

Security Associations (SAs)

A **Security Association** (SA) is a secure communication channel and its operating parameters, such as the encryption algorithm, keys and lifetime. There are two SA negotiation phases within IKE—Phase 1 and Phase 2. The general flow of the IKE protocol is as follows:

1. **IKE Phase 1 (Establish an IKE SA)**

    The purpose of IKE Phase 1 is to establish an IKE SA, which is a secure, encrypted communication channel used for further IKE communication. During Phase 1 negotiations, the IKE peers authenticate the identity each other and generate a **Diffie-Hellman** shared value (described in "Generating Shared Keys: Diffie-Hellman" (page 25).) that is used as the base for shared keys.

    IKE can use one of two methods, or exchange types, to establish the IKE SA:
    *   **Main Mode**: In Main Mode negotiations, the IKE peers select IKE parameters (configured in IKE policies) based on the remote system's IP address in the IP packet header. The IKE peers exchange ID information after they establish a secure, encrypted communication channel.
    *   **Aggressive Mode**: In Aggressive Mode negotiations, the IKE initiator sends ID information in the first packet. This enables the IKE responder to select IKE parameters, such as the encryption information, based on ID information instead of the IKE peer's IP address extracted from the IP packet header. Aggressive Mode is quicker and requires the peers to exchange fewer packets, but is less secure because the peers exchange identity information in clear text.

    The IKE protocol specification requires Main Mode support; Aggressive Mode support is optional. Aggressive Mode is required when IKE is used with autoconfiguration clients and Mobile IPv6 clients because these clients do not have fixed IP addresses. Aggressive Mode enables IKE to select IKE parameters without using the remote address in the IP packet header.

Most IPsec IKEv1 implementations, including IPsec, use Main Mode by default.

The IKE Phase 1 negotiation is also referred to as a Main Mode (MM) or an Aggressive Mode (AM) negotiation, depending on the exchange type used.

2. IKE Phase 2 (Establish IPsec SAs)

Using the secure communication channel provided by the IKE SA, IKE negotiates IPsec SAs. An IPsec SA is a security association used to exchange IPsec ESP or AH packets. The IPsec SA operating parameters include the IPsec protocol used (ESP or AH), the mode (transport or tunnel), the cryptographic algorithms (such as AES and SHA-1), the cryptographic keys, the SA lifetime, and the endpoints (IP addresses, protocol and port numbers).

IPsec SAs are unidirectional, so each Phase Two negotiation negotiates two IPsec SAs: one for inbound packets from the remote endpoint and one for outbound packets to the remote endpoint.

The IKE Phase 2 negotiation is also referred to as a **Quick Mode (QM) negotiation**.

Generating Shared Keys: Diffie-Hellman

IKE and IPsec SAs use shared keys to encrypt and authenticate communication. To be effective, a shared key must be kept private, so other parties cannot decrypt the data or generate a valid authentication code for modified data. This creates a challenge: How do the two parties agree on the same shared key? How can you distribute the same key to both parties without exposing it to other parties listening on the network?

One method for distributing shared keys is to use the **Diffie-Hellman** algorithm to dynamically generate shared keys. The Diffie-Hellman algorithm enables two parties to establish a shared, secret value while exchanging information over a nonsecure channel.

The Diffie-Hellman algorithm is based on the principle that $(x^a)^b$ and $(x^b)^a$ are both equivalent to $x^{(a*b)}$. With Diffie-Hellman key generation, each party generates two numbers: one public and one private. These values are based on a selected, well-known numeric base, or **Diffie-Hellman group** . The two parties first select the same Diffie-Hellman group. The two parties each select a public value and generate a mathematically related private value. The two parties exchange public values. This exchange can occur via a nonsecure channel. Each party then uses its private value and the other party's public value to generate a new value. Because of the mathematical properties of the numbers, each party generates the same value, which can then be used as a shared key or use as a base value to generate multiple shared keys.

IKE Primary Authentication

Diffie-Hellman is vulnerable to **third-party attacks**, in which a third party intercepts messages between two attacked parties, A and B. A and B assume they are exchanging messages with each other, but are exchanging messages with the third party. The attacker assumes the identity of A to exchange messages with B, and assumes the identity of B to exchange messages with A.

Because of this vulnerability, IKE must authenticate the identities of the parties using the Diffie-Hellman algorithm. This process is known as **IKE primary authentication**.

OpenVMS IPsec supports two IKE primary authentication methods:
- Preshared keys
- Digital Signatures

> **NOTE:** Digital signatures are not supported in this EAK release.

IKE Preshared Key Authentication

With preshared key authentication, you must manually configure the same, shared on both systems—a **preshared key**.

The two parties establish a shared key (the preshared key) prior to the Diffie-Hellman exchange using an **out-of-band key exchange**, or a key exchange that does not use normal computer communication channels, such as a face-to-face meeting or telephone call where the two parties agree on a key. The preshared key is used only for the primary authentication. The two negotiating entities then generate dynamic shared keys for the IKE SAs and IPsec SAs.

Preshared keys do not require a Certificate Authority or Public Key Infrastructure.

IKE Digital Signature Authentication

Digital signatures are based on **security certificates** , and are managed using a **Public Key Infrastructure (PKI)**. To use certificates with IPsec the PKI must support the following certificate file formats and access methods:

- Certificate Requests: The CA must support Certificate Signing Requests (CSRs) in Public Key Cryptography Standards (PKCS) Certification Request Syntax #10 format (commonly referred to as PKCS#10) and encoded using Privacy-Enhanced Mail (PEM) base64 encoding. This CSR format is typically used for "copy and paste" certificate requests.
- Certificates: The CA must provide X.509 Version 3 certificates encoded using base64 encoding (sometimes referred to as base64 PEM format).
- Certificate Revocation Lists: The CA must provide X.509 Version 1 or X.509 Version 2 Certificate Revocation Lists formatted using Distinguished Encoding Rules (DER).

IKE Automatic Re-keying

The IKE protocol also enables IPsec to dynamically negotiate new IPsec keys rather than exposing the same key for long periods. You can configure key lifetimes based on time or number of bytes sent.

Manual Keys

Manual keys are an alternative to IKE. Instead of using IKE to dynamically generate and distribute cryptography keys for ESP and AH, the cryptography keys are static and manually distributed using an out-of-band key exchange. Because manual keys are static, using them is less secure than using IKE. Manual keys are typically used only when the remote system does not support IKE, such as a Mobile IPv6 node that does not support IKE.

OpenVMS IPsec Topologies

You can use IPsec between hosts (end nodes), between gateways, and between a host and a gateway in an IP network. You can install OpenVMS IPsec only on end nodes. An OpenVMS IPsec system can have the following roles:

- A host in a host-to-host IPsec topology
- A host in a host-to-gateway IPsec topology
- A host in a host-to-host IPsec tunnel topology, frequently referred to as an end-to-end tunnel. End-to-end tunnels are commonly used in iSCSI topologies.

Uses for OpenVMS IPsec include:

- Providing host-to-host security within an intranet. You can use OpenVMS IPsec to secure intranet packets that carry sensitive data, such as personnel and payroll information.
- Creating VPNs to allow external partners to access selected internal systems through the public Internet.
- Protecting backend servers in topologies that external clients access through application gateway servers in an area outside corporate firewalls (demilitarized zone, or DMZ).

Host-to-Host Security Within an Internal Network

Two end hosts can run OpenVMS IPsec locally to protect communication between them, with or without intermediate gateways.

You can use OpenVMS IPsec to secure sensitive network communication within an enterprise, such as network communication for Human Resources (HR) or payroll groups. Host-to-host IPsec secures all packets within the HR subnet, and between node E1 in the engineering subnet and H1 in the HR subnet.

Host-to-Host VPN Across the Internet

IPsec can provide secure VPN tunnels through the public Internet. VPN tunnels protect packet transfer from a remote workstation to a corporate intranet or link geographically dispersed portions of an intranet without using expensive leased lines. VPN tunnels can also link the computing facilities of business partners and secure mobile and wireless node communications.

The supplier and manufacturer have separate intranets that are connected to the public Internet using Internet Service Providers (ISPs). System A on the supplier's intranet and System B on the manufacture's subnet communicate with a host-to-host IPsec topology. For added security, you can configure filtering on the manufacturer's firewall so that it checks the traffic to and from system A and allows only IPsec packets between system A and B to pass.

Host-to-Gateway VPN Across the Internet

You can also use IPsec to create a host-to-gateway VPN across the Internet. The manufacturer's IP router is an IPsec gateway, and system A establishes the IPsec session with the manufacturer's router.

In this example, system A can easily access all systems in the manufacturer's network; therefore you **must** configure filtering on the manufacturer's firewall to check the traffic to and from system A and allow only IPsec packets between system A and B to pass. In addition, packets between the router and system B are not secured.

In the host-to-gateway VPN topology, OpenVMS IPsec is used on system A. The router uses an IPsec gateway product provided by another vendor.

Application Server in DMZ with Back-End Server

More enterprises are putting application servers in a "**demilitarized zone (DMZ)**"—that is, outside corporate firewalls—for business partners or public access. Because inbound connections from the Internet are allowed to these servers, they are vulnerable to attack. In many cases, the

application servers in the DMZ are configured as application gateways, or proxy servers, that open a second connection to backend servers within the internal network and forward client requests to the back-end servers.

In these scenarios, OpenVMS IPsec can secure the host-to-host data path between the gateway application server in the   DMZ and the backend server. You must configure filtering on the gateway application server (B) to limit access to the backend servers.

Securing Access between the Client and DMZ Server

For added security, you can use IPsec between the client and the gateway application server in the DMZ. Alternatively you can deploy an IPsec VPN gateway appliance on the external network. The IPsec VPN gateway appliance and the gateway application server in the DMZ establish IPsec gateway-to-gateway sessions. Client requests can go through the external IPsec VPN gateway appliance to the gateway application server in the DMZ and then to the backend server. The IPsec VPN gateway enables clients to access the backend servers without having IPsec locally installed.

2 Enabling and Configuring OpenVMS IPsec

This chapter describes enabling and configuring OpenVMS IPsec.

Enabling OpenVMS IPsec

To enable the IPsec service, enter the following command:

`@SYS$MANAGER:TCPIP$CONFIG`.

At the main menu, select the following:

4 (`Optional components`), then select 6 (`Configure IPsec`).

Finally, select 1 (`Enable service on this node`).

`TCPIP$CONFIG` creates a directory for the TCPIP$IPSEC account on TCPIP$IPSEC_DEVICE which defaults to SYS$SPECIFIC.

Starting Up and Shutting Down IPsec

To start up and shut down OpenVMS IPsec, use the appropriate option displayed:

Select 6 (`Configure IPsec`).

Alternately, you can perform the function interactively:

`@SYS$STARTUP:TCPIP$IPSEC_STARTUP`

or

`@SYS$STARTUP:TCPIP$IPSEC_SHUTDOWN`

> **NOTE:** The command `STOP /ID=n` or `STOP TCPIP$IPSEC_1` does not cause the Policy Manager exit handler to be called; therefore, the execlet stops responding. To exit correctly, use the `TCPIP$IPSEC_SHUTDOWN.COM` procedure. If you must shut down manually, use the `STOP /IMAGE /ID=n` command.

Logical Names

The following table lists logical names used in OpenVMS IPsec

Logical Name	Function
TCPIP$IPSEC_DEVICE	If not defined in an OpenVMS startup routine, defaults to SYS$SPECIFIC.
TCPIP$IPSEC_HOME	The location where TCPIP$IPSEC_RUN.LOG files are created. Also several temporary files are created, including IPSEC_SYSCONFIG_OUTPUT.LIS and IPSEC_SYSCONFIG_TEMP.DAT, which are used to coordinate startup of the PM with the status of IPSEC execlet loaded with sysconfig.
TCPIP$IPSEC_PM_CONFIG_PORT	The port number that the Policy Manager and the ipsec_config utility will use for communication.

Configuring IPsec with ipsec_config

> **NOTE:** The default method for configuring the IPsec Policy Manager (PM) will be through the use of the `ipsec_config`. A secondary method also exists, via an XML file, `tcpip$ipsec_home:quicksec.xml`. If this file exists, then it will be used and `ipsec_config` configuration will fail. Note that this secondary method of configuration may only exist for the EAK release.

Using `ipsec_config`

The `ipsec_config` utility adds, deletes, and displays OpenVMS IPsec configuration objects stored in the configuration database, `tcpip$ipsec_home:ipsec_config.db`. If the IPsec Policy Manager (PM) is running, `ipsec_config` also adds and deletes configuration information in the runtime policy database. The `ipsec_config` utility supports the following commands:

- **`ipsec_config add`**
- **`ipsec_config batch`**
- **`ipsec_config delete`**
- **`ipsec_config show`**

📝 **NOTE:** When you enable the IPsec service with `TCPIP$CONFIG`, if it does not exist, an SQLite database is created in `TCPIP$IPSEC_HOME` as `IPSEC_CONFIG.DB`. This file acts as a backup to the dynamic information stored in the runtime policy database of the PM. Information in this file is retained across reboots and it is read at PM startup. The file is not human-readable.

To use the ipsec_config utility, define the symbol (foreign command) by executing the following command:

`@sys$manager:tcpip$define_commands.com`

When an `ipsec_config` command is issued, the following actions occur:

- The backup SQLite database is updated (`tcpip$ipsec_home:ipsec_config.db`).
- A TCP connection is established with the PM (port 81 by default) and the command is sent to the PM to accomplish a dynamic policy update.

At PM startup, the PM reads `tcpip$ipsec_home:ipsec_config.db` and applies any policy updates contained therein.

Duplicate entries, or the issuance of `ipsec_config` ADD commands for entries of the same *name*, will not succeed.

When `ipsec_config` initialization occurs in the PM, the PM log file `SYS$SPECIFIC:[TCPIP$IPSEC]TCPIP$IPSEC_RUN.LOG` contains the following line:

`tcpip$ipsec_pm.exe: Initializing ipsec_config communication`

Logical Names

You can redefine the communications port used between the `ipsec_config` utility and the PM by defining the logical name `TCPIP$IPSEC_PM_CONFIG_PORT` in the system logical name table (/system).

`ipsec_config add`

The `ipsec_config add` command adds objects to the configuration database. For example, the following command adds a host IPsec policy to the configuration database.

```
ipsec_config add host my_host_policy -source 10.1.1.1 -
-destination 10.0.0.0/8/TELNET -pri 100 -
-action ESP_AES128_HMAC_SHA1
```

`ipsec_config batch`

The `ipsec_config batch` command allows you to use `ipsec_config` in batch mode. In batch mode, `ipsec_config` reads add and delete operations from a file. Batch mode allows administrators to add and delete multiple configuration objects in one operation. This is useful if you are adding or deleting configuration records that affect other operations.

HP recommends that you use a batch file to add configuration information. A batch file provides a permanent record of the configuration data and can be used to re-create the configuration database.

Batch File Processing

The `ipsec_config` utility processes the operations in a batch file as a group. If one operation is invalid, all operations in the batch file fail. The `ipsec_config` utility first verifies each operation in the batch file for syntax errors and collisions (object names and priority values) with existing entries in the configuration database. If all operations in the batch file are valid, the IPsec infrastructure updates the configuration database with all operations at the same time. If the IPsec Policy Manager is active and running, the `ipsec_config` utility also updates the runtime policy database.

Batch File Syntax

The syntax for add and delete operations in `ipsec_config` batch files is the same as the syntax for `ipsec_config` add and `ipsec_config` delete commands, but without the leading `ipsec_config` command name. For example, the following entry is a valid add operation for a batch file:

```
add host my_host_policy -source 10.1.1.1 \
-destination 10.0.0.0/8/TELNET -pri 100 \
-action ESP_AES128_HMAC_SHA1
```

Comments

Lines starting with a pound sign (#) are interpreted as comments. The continuation character to be used in a batch file is the backslash (\) Comment lines within an operation are not allowed.

ipsec_config delete

The `ipsec_config delete` command deletes objects from the configuration and runtime databases. For example, the following command deletes the host IPsec policy `my_host_policy` from the configuration database:

```
ipsec_config delete host my_host_policy
```

ipsec_config show

The `ipsec_config show` command displays objects in the configuration database. For example, the following command displays the host IPsec policies in the configuration database:

```
ipsec_config show host
```

The `ipsec_config show all` command displays the entire contents of the database.

Profile File

An `ipsec_config` profile file contains default argument values that are evaluated in `ipsec_config` add commands if the user does not specify the values in the command. The values are evaluated once, when the policy is added to the configuration database. Values used from the profile file become part of the configuration record for the policy.

You can specify a profile file name with the `-profile` argument as part of an `ipsec_config` command. By default, `ipsec_config` uses the `tcpip$ipsec_home:ipsec_profile.dat` profile file, which is shipped with OpenVMS IPsec. In most topologies, you can use the default values supplied in the `tcpip$ipsec_home:ipsec_profile.dat` file.

OpenVMS IPsec also has internal default values that are the same as the values in the `tcpip$ipsec_home:ipsec_profile.dat` file shipped with the product. If the `tcpip$ipsec_home:ipsec_profile.dat` file does not exist and the user does not specify an alternate profile file, OpenVMS IPsec uses its internal default values.

Using a Profile File with a Batch File

You can specify the `profile` argument as part of the `ipsec_config` batch command line and `ipsec_config` will apply it to all entries in the batch file. The `profile` argument is illegal inside batch files (you cannot specify the `profile` argument as part of a statement inside a batch file).

Profile File Structure

The profile file is separated into sections that contain default parameter values for different configuration objects. For example, the HostPolicy-Defaults section contains defaults for host IPsec policies, which are created using the `ipsec_config add host` command. Each section is delimited by BEGIN and END statements.

Creating a Customized Profile File

In most topologies, you can use the default values in `tcpip$ipsec_home:ipsec_profile.dat`. If you want to create a customized profile file, make a copy of the `ttcpip$ipsec_home:ipsec_profile.dat` file and edit the copy with a text editor.

You may want to create a customized profile file to change the default source address parameter (`source` parameter) in the following topologies:

- IPv6 networks
- Multihomed nodes with private interfaces

The default source address parameter values in `tcpip$ipsec_home:ipsec_profile.dat` are `0.0.0.0/0/0` (IPv4 address 0.0.0.0, address prefix length 0, port 0). This matches any IPv4 address and any port number. In most topologies, this is appropriate since the default source (local) address will be any IPv4 address on the local system.

IPv6 Networks

If you have a network that primarily contains IPv6 nodes, you can change the `source` parameter value to match any IPv6 address and any port number (`0:0/0/0`) in the HostPolicy-Defaults section of the profile file. You can also change the `remote` parameter value in the IKEPolicy-Defaults section to match any IPv6 address (`0::0/0`).

Multihomed Nodes with Private Interfaces

If the local system is multihomed with one public IP interface and one or more private IP interfaces, you may want to secure only the one public IP interface. In this case, you can set the default `source` parameter value to the address of the public IP interface in the HostPolicy-Defaults section of the profile file.

Dynamic Configuration Updates

The `ipsec_config` utility dynamically updates the configuration database. If IPsec is running, `ipsec_config` also updates the runtime IPsec policy database, and runtime IKE configuration data (IKE policies and authentication records).

Dynamic Deletions

If you delete an object while the IPsec Policy Manager is running, the IPsec Policy Manager deletes it from its runtime database. If you delete an IPsec policy, the IPsec Policy Manager deletes any associated IPsec SAs. If you delete an IKE policy, OpenVMS deletes any associated IKE SAs. IPsec SAs negotiated using the IKE SAs may continue to operate, but IKE peers will be unable to send control messages for the affected IPsec SAs.

`nocommit` Argument

The `nocommit` argument validates entries but does not update the configuration and runtime databases. The `nocommit` argument is illegal inside batch files (you cannot specify the `nocommit` argument as part of a statement inside a batch file). You can specify the `nocommit`argument as part of the `ipsec_config` batch command line and `ipsec_config` will apply it to all entries in the batch file.

Configuration Overview

There are seven main configuration components:

- **Host IPsec Policies**

  Host IPsec policies specify OpenVMS IPsec behavior for IP packets sent or received by the local system as an end host. A host IPsec policy contains address specifications used to select the host IPsec policy for a packet. A host IPsec policy also specifies the OpenVMS IPsec behavior (action) for packets using the policy: pass the packets in clear text, discard the packets, or apply an IPsec transform (AH or ESP) to the packets.

- **Tunnel IPsec Policies**

  Tunnel IPsec policies specify the behavior for tunnel endpoints. If the local system is an end host in a end-to-end tunnel (host-to-host tunnel) topology, or the end host in a host-to-gateway tunnel topology, you must configure tunnel IPsec policies. If the local system is only an end host with no IPsec tunneling, do not configure tunnel IPsec policies.

- **IKE Policies**

  An IKE policy defines the parameters used when negotiating an IKE Security Association (SA). IPsec uses IKE SAs to negotiate IPsec SAs; an IKE SA must exist with a remote system before IPsec can negotiate IPsec SAs.

- **IKE Authentication Records**

  IKE Authentication records contain information that IKE uses to authenticate identities with the remote system, including local and remote ID values, exchange mode, and preshared keys, if preshared keys are used. You *must* configure IKE authentication records if you use preshared keys for IKE authentication.

- **Security Certificates**

  You can use security certificates with RSA signatures for IKE authentication (also referred to as primary authentication) instead of preshared keys.

📝 **NOTE:** The EAK does not support the use of Security Certificates.

You must configure the above components in a specific order, therefore HP recommends that you use the following procedure to configure IPsec:

1. Configure host IPsec or tunnel IPsec policies.

   See "Step 1: Configuring Host or Tunnel IPsec Policies" (page 37) for a description of this step.

2. Configure IKE policies.

   See "Step 2: Configuring IKE Policies" (page 43) for a description of this step. Skip this step if the local system uses only manual keys for IPsec.

3. Configure IKE preshared keys using authentication records.

   See "Step 3: Configuring Preshared Keys Using Authentication Records" (page 46) for a description of this step. Skip this step if the local system uses only manual keys for IPsec.

4. Configure security certificates and ID information, if you are using RSA signatures for IKE authentication.

📝 **NOTE:** Security Certificates are not supported in this EAK.

5. Verify the batch file.

   HP recommends that you use an `ipsec_config` batch file to add configuration information, and that you use the `ipsec_config batch` command with the `nocommit` option to verify

the contents of the batch file before committing the batch file operations to the database file. See "Step 5: Verifying the Batch File Syntax" (page 53) for a description of this step.

6. Commit the batch file operations to the database and start IPsec to verify operation.

   After you have verified the contents of the batch file, commit the batch file operations to the configuration database file. Start IPsec and verify operation. See "Step 6: Committing the Batch File Configuration and Verifying Operation" (page 54) for a description of this step.

Step 1: Configuring Host or Tunnel IPsec Policies

Host IPsec policies specify IPsec behavior for IP packets sent or received by the local system as an end host. Each host IPsec policy includes address specifications used to select the host IPsec policy for a packet, and the action for packets using the policy: pass the packets in clear text, discard the packets, or apply an IPsec transform (AH or ESP) to the packets.

If the host policy is for an end host in an end-to-end tunnel (host-to-host tunnel) topology or and end host in a host-to-gateway topology, the host policy must specify the `-tunnel` option.

Tunnel IPsec policies specify IPsec behavior for IP packets tunneled by the local system. In an IPsec tunnel, a tunnel endpoint system encapsulates the original packet in a new IPsec packet with an AH or ESP header. The other tunnel endpoint system processes the AH or ESP header, decapsulates the packet, and sends the packet to the destination address in the original packet header.

Host Policy Order and Selection

When an IPsec system sends a packet or receives a packet for an address on the local system, IPsec searches the host IPsec policies according to the value of the `priority` parameter for each policy and selects the first policy with address, protocol and port specifications that match the packet. OpenVMS IPsec then takes the action specified in the selected host IPsec policy.

`default` Host IPsec Policy

The IPsec configuration database includes a host IPsec policy named `default`. IPsec uses the `default` host IPsec policy for a packet if no other host IPsec policies match the packet. The `default` host IPsec policy allows packets to pass in clear text. You cannot delete the `default` host IPsec policy, or modify any argument values except the value for the behavior (the `action` argument). Use the following command to change the `default` host IPsec policy so it discards (or drops) packets:

```
ipsec_config add host default -action DISCARD
```

You may also change the default host IPsec policy so that it rejects packets. Reject is similar to discard except that with reject the Policy Manager sends back a `TCP RST`(reset). Use the following command to change the default host IPsec policy so it rejects packets:

```
ipsec_config add host default -action REJECT
```

To change back the behavior of the `default` host IPsec policy to pass packets in clear text, use the following command:

```
ipsec_config add host default -action PASS
```

Automatic Priority Decrement

There are two ways to set the priority of a host policy:
- Specify the `priority` argument to explicitly set the priority.
- Omit the `priority` argument and have `ipsec_config` assign a priority using the automatic priority decrement value so that the new policy is the last policy evaluated before the `default` policy.

If you omit the priority argument, `ipsec_config` assigns a priority value that is set to the current lowest priority value for host policies (lowest priority) in the configuration database, decremented by the automatic priority decrement value for host policies. The result is that the new policy is the last host policy evaluated before the default policy. The automatic priority decrement value for the host policies defaults to 2.

`ipsec_config add host` Syntax

If you are not using manual keys, you can use the following `ipsec_config add host` syntax in most installations :

```
ipsec_config add host host_policy_name
[-source ip_addr[/prefix[/port_number|service_name]]]
[-destination ip_addr[/[prefix[/port_number|service_name]]]
[-protocol protocol_id] [-priority priority_number]
[-action PASS|DISCARD|REJECT|transform_list]
```

HP recommends that you use an ipsec_config batch file to configure IPsec. To specify an `add host` operation for an `ipsec_config` batch file, use the above syntax without the `ipsec_config` command name:

```
add host host_policy_name
[-source ip_addr[/[prefix][/port_number|service_name]]]
[-destination ip_addr[/prefix][/port_number|service_name]]]
[-protocol protocol_id] [-priority priority_number]
[-action PASS|DISCARD|REJECT|transform_list]
```

The complete `ipsec_config add host` syntax specification also allows you to specify the following arguments:

- `nocommit` (verify the syntax but do not commit the information to the database)
- `profile` (alternate profile file)
- `in` and `out` (inbound and outbound SA information for manual keys)

host_policy_name

The *host_policy_name* is the user-defined name for the host IPsec policy. This name must be unique for each host IPsec policy and is case-sensitive.

Acceptable Values: 1 - 63 characters. Each character must be an ASCII alphanumeric character, hyphen (-), or underscore (_).

The name **default** is reserved. See "default Host IPsec Policy" (page 37) for more information.

-source and -destination *ip_addr [/prefix [/port_number |service_name]]*

IPsec uses the *ip_addr*, *prefix*, and *port_number* or *service_name* with the `protocol` argument to form an address filter. IPsec uses the address filter to select an IPsec policy for a packet. Specify a local IP address for the source *ip_addr*. For an outbound packet, IPsec compares the source address filter with the source address fields in the packet, and the destination address filter with the destination address fields in the packet. For an inbound packet, IPsec compares the source address filter with the destination address fields in the packet, and the destination address filter with the source address fields in the packet.

> **TIP:** For host policies, the source address is the local address and the destination address is the remote address.

Default: If you do not specify *ip_addr*, *prefix*, and *port_number* or *service_name*, `ipsec_config` uses the value of the `source` or `destination` parameter in the HostPolicy-Defaults section of the profile file used. The default value for `source` and `destination` is `0.0.0.0/0/0` (match any IPv4 address, any port).

ip_addr

The *ip_addr* is the source or destination IP address.

Acceptable Values: An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. The IP address type (IPv4 or IPv6) must be the same for the source and destination address. IPsec does not support unspecified IPv6 addresses. However, you can

use the double-colon (::) notation within a specified IPv6 address to denote a number of zeros (0) within an address. The address cannot be a broadcast, subnet broadcast, multicast, or anycast address.

prefix

The *prefix* is the prefix length, or the number of leading bits that must match when comparing the IP address in a packet with *ip_addr*.

For IPv4 addresses, a prefix length of 32 bits indicates that all the bits in both addresses must match. This prefix length is equivalent to an address mask of 255.255.255.255. Use a value less than 32 to specify a subnet address filter.

For IPv6 addresses, a prefix length of 128 bits indicates that all the bits in both addresses must match. Use a value less than 128 to specify a subnet address filter.

Range: 0 - 32 for an IPv4 address; 0 - 128 for an IPv6 address. If you are using manual keys, prefix must be 32 if *ip_addr* is an IPv4 address or 128 if *ip_addr* is an IPv6 address.

Default: 32 if *ip_addr* is a non-zero IPv4 address, 128 if *ip_addr* is a non-zero IPv6 address, or 0 (match any address) if *ip_addr* is an all-zeros address (0.0.0.0 or 0::0). You must specify a prefix value if you specify a port or service name as part of the address filter.

port

The *port* is the upper-layer protocol (TCP or UDP) port number Specify the upper-layer protocol with the protocol argument described below.

Acceptable Values: 0 - 65535. 0 indicates all ports. The upper-layer protocol must be TCP or UDP if you specify a non-zero port number.

Default: 0 (all ports).

service_name

The *service_name* is a character string that specifies a network service. The ipsec_config utility will add a policy to the configuration database with the appropriate port number and protocol, as listed below. You cannot specify *service_name* and protocol gin the same policy.

Table 2-1 ipsec_config **Service Names**

Service Name	Port	Protocol
DNS-TCP	53	TCP
DNS-UDP	53	UDP
FTP-DATA	20	TCP
FTP-CONTROL	21	TCP
HTTP-TCP	80	TCP
HTTP-UDP	80	UDP
NTP	123	UDP
REXEC	512	TCP
RLOGIN	513	TCP
SMTP	25	TCP
TELNET	23	TCP
TFTP	69	UDP

-protocol *protocol_id*

The *protocol_id* is the value or name of the upper-layer protocol that OpenVMS IPsec uses in the address filter to select an IPsec policy for a packet. You cannot specify `protocol` and a *service_name* in the same policy.

Specifying ICMPV6 affects only the following ICMPv6 messages: Echo Request, Echo Reply, Mobile Prefix Solicitation, Mobile Prefix Advertisement.

To ensure proper operation of IPv6 networks, OpenVMS IPsec always allows all ICMPv6 messages not listed above to pass in clear text

Acceptable Values: Integer value 0 (any protocol) - 255, or one of the following protocol names:

- **TCP**
- **UDP**
- **ICMP**
- **ICMPV6**
- **IGMP**
- **ALL** (any protocol)

The protocols **ICMP** and **IGMP** are valid with IPv4 addresses only. The protocol **ICMPV6** is valid with IPv6 addresses only.

The *protocol_id* must be TCP or UDP if *port* is non-zero.

Default: ALL .

> **NOTE:** The EAK does not currently support the use of IPv6 addresses

> **CAUTION:** Discarding or requiring ICMP messages for IPv4 (protocol value 1) to be encrypted or authenticated may cause connectivity problems.

-priority *priority_number*

The *priority_number* is the priority value OpenVMS IPsec uses when selecting a host IPsec policy (a higher priority value has a higher priority). The priority must be unique for each host IPsec policy.

Range: 99999999 - 16.

Default: If you do not specify a priority, `ipsec_config` assigns a priority value that is set to the current lowest priority value (lowest priority) for host IPsec policies in the configuration database, decremented by the automatic priority decrement value for host IPsec policies. The default automatic priority decrement value is 2.

-tunnel

If packets using this host IPsec policy will be tunneled and the local system is one of the tunnel endpoints, you must indicate this by using the *tunnel* argument. The argument does not take a value.

-action

The `action` argument specifies the action OpenVMS IPsec will perform on packets using this policy.

Default: The default definition for `action` is DISCARD.

PASS

Allow packets using this host IPsec policy to pass in clear text with no alteration. The default host IPsec policy shipped with the product specifies -action PASS .

```
DISCARD
```

Discard (silently drop) packets using this host IPsec Policy.

```
REJECT
```

Reject (drop but send a TCP RST (reset)) packets using this host IPsec Policy.

transform_list

A transform specifies the IPsec authentication and encryption applied to packets using AH (Authentication Header) and ESP (Encapsulation Security Payload) headers. A transform list specifies the transforms acceptable for packets using the policy. The OpenVMS IPsec Policy Manager proposes the transform list when negotiating the transform for IPsec Security Associations (SAs) with a remote system.

The transform list in a host policy are transport transforms and are applicable to the host-to-host SA (end-to-end or transport SA) between the source and destination addresses.

If you are using dynamic keys, the transform list can contain:

- A list that contains up to 2 AH transforms
- A list that contains up to 8 ESP transforms.
- A list that contains one nested AH and ESP transform (ESP nested inside of AH)

Use a comma to separate multiple transform specifications.

The order of transforms in the transform list is significant. The first transform is the most preferable and the last transform is the least preferable. At least one transform must match a transform configured on the remote system.

The format for each transform is:

transform_name [/*lifetime_seconds* [/*lifetime_kbytes*]]

Where:

transform_name

The transform_name is one of the following AH (Authentication Header) or ESP (Encapsulation Security Payload) transform specifications, or a   nested AH and ESP transform formed by joining an AH transform and an ESP transform with a plus sign (+). For example, AH_MD5+ESP_3DES_HMAC_SHA1.

📝 **NOTE:** *lifetime_seconds* and *lifetime_kbytes* are read only from the first transform in the list.

💡 **TIP:** AES128 is the most secure form of encryption, with performance comparable to or better than DES and 3DES.

Table 2-2 `ipsec_config` **Transforms**

Transform Name	Description
`AH_MD5`	AH, with 128-bit key Hashed Message Authentication Code using RSA Message Digest-5, HMAC-MD5.
`AH_SHA1`	AH, with 160-bit key HMAC using Secure Hash Algorithm-1, HMAC-SHA1.
`ESP_AES128_HMAC_MD5`	ESP with 128-bit Advanced Encryption Standard (AES128) CBC, authenticated with HMAC-MD5.
`ESP_AES128_HMAC_SHA1`	ESP with 128-bit Advanced Encryption Standard (AES128) CBC, authenticated with HMAC-SHA1.

Table 2-2 ipsec_config Transforms *(continued)*

Transform Name	Description
`ESP_DES_HMAC_MD5`	ESP with 56-bit Data Encryption Standard, Cipher Block Chaining Mode (DES), authenticated with HMAC-MD5.
`ESP_DES_HMAC_SHA1`	ESP with 56-bit Data Encryption Standard, Cipher Block Chaining Mode (DES), authenticated with HMAC-SHA1.
`ESP_3DES_HMAC_MD5`	ESP with triple-DES CBC, three encryption iterations, each with a different 56-bit key (3DES), authenticated with HMAC-MD5.
`ESP_3DES_HMAC_SHA1`	ESP with triple-DES CBC, three encryption iterations, each with a different 56-bit key (3DES), authenticated with HMAC-SHA1.
`ESP_NULL_HMAC_MD5`	ESP with null encryption and authenticated with HMAC-MD5.
`ESP_NULL_HMAC_SHA1`	ESP with null encryption and authenticated with HMAC-SHA1.

lifetime_seconds

The *lifetime_seconds* is the maximum lifetime for the IPsec SA, in seconds. A transform lifetime can be specified by time (seconds), and by kilobytes transmitted or received. IPsec considers the lifetime to be exceeded if either value is exceeded.

Range: 0 (default), 30 - 4294967294 seconds (approximately 497102 days).

Default: 28,800 (8 hours).

lifetime_kbytes

The *lifetime_kbytes* is the maximum lifetime for the IPsec SA, measured by kilobytes transmitted or received. A transform lifetime can be specified by time (seconds), and by kilobytes transmitted or received. IPsec considers the lifetime to be exceeded if either value is exceeded.

Range: 0 (default), 500 - 4294967294 kilobytes.

Default: 0.

Step 2: Configuring IKE Policies

You must specify configure an IKE policy if you are using dynamic keys for IPsec.

IPsec uses the parameters in an IKE policy when using the IKE protocol to establish IKE Security Associations (SAs) with remote systems. IPsec uses IKE SAs to negotiate IPsec SAs; an IKE SA must exist with a remote system before IPsec can negotiate IPsec SAs.

You must specify an IKE policy if you are using dynamic keys for IPsec.

IKE Policy Order and Selection

The OpenVMS IPsec Policy Manager stores the IKE policy information alongside the associated Host IPsec Policy information. The Host IPsec Policy is retrieved to obtain or set the IKE policy information.

`ipsec_config add ike` Syntax

You can use the following `ipsec_config add ike` syntax in most installations:

```
ipsec_config add ike ike_policy_name
-remote ip_addr[/prefix]
[-authentication PSK|RSASIG]
[-hash MD5|SHA1] [-encryption AES|DES|3DES]
[-life lifetime_seconds]
```

HP recommends that you use an ipsec_config batch file to configure IPsec. To specify an `add ike` operation for an `ipsec_config` batch file, use the above syntax without the `ipsec_config` command name:

```
add ike ike_policy_name -remote ip_addr[/prefix]
[-authentication PSK|RSASIG]
[-hash MD5|SHA1] [-encryption AES|DES|3DES]
[-life lifetime_seconds]
```

The complete `ipsec_config add ike` syntax specification also allows you to specify the following arguments:

- `nocommit` (verify the syntax but do not commit the information to the database)
- `profile` (alternate profile file)

ike_policy_name

The *ike_policy_name* is the user-defined name for the IKE policy. This name must match the name given to the host policy with which the IKE policy is associated.

Acceptable Values: 1 - 63 characters. Each character must be an ASCII alphanumeric character, hyphen (-), or underscore (_).

-remote *ip_addr* [/*prefix*]

The *ip_addr* and *prefix* are the IP address and network prefix length that specifies the remote system or subnet for this policy. HP recommends that you do not specify a wildcard address (0.0.0.0/0 or 0::0/0). Wildcard addresses allow unauthorized systems to engage the local systems in IKE negotiations.

Where:

ip_addr

The *ip_addr* is the remote IP address.

Acceptable Values: An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. The IP address type (IPv4 or IPv6) must be the same for the source and destination address. IPsec does not support unspecified IPv6 addresses. However, you can

use the double-colon (::) notation within a specified IPv6 address to denote a number of zeros (0) within an address. The address must be a unicast address.

Default: None.

prefix

The *prefix* is the prefix length, or the number of leading bits that must match when comparing the remote IP address with *ip_addr*.

For IPv4 addresses, a prefix length of 32 bits indicates that all the bits in both addresses must match. This prefix length is equivalent to an address mask of 255.255.255.255. Use a value less than 32 to specify a subnet address filter.

For IPv6 addresses, a prefix length of 128 bits indicates that all the bits in both addresses must match. Use a value less than 128 to specify a subnet address filter.

Range: 0 - 32 for an IPv4 address; 0 - 128 for an IPv6 address. If you are using manual keys, prefix must be 32 if *ip_addr* is an IPv4 address or 128 if *ip_addr* is an IPv6 address.

Default: 32 if *ip_addr* is a non-zero IPv4 address, 128 if *ip_addr* is a non-zero IPv6 address, or 0 (match any address) if *ip_addr* is an all-zeros address (0.0.0.0 or 0::0).

-authentication *authentication_type*

The authentication_type is the primary authentication method IPsec will use when establishing the IKE SA. This must match the method configured on the remote system.

Acceptable Values:

PSK (preshared key)
RSASIG (RSA signatures using security certificates)

📝 **NOTE:**   The EAK release does not currently support Security Certificates.

If you specify PSK, you must configure a preshared key using the ipsec_config add auth command. If you specify RSASIG, you must use security certificates.

Default: The default authentication parameter value is PSK.

-group 1|2

The group argument specifies the Diffie-Hellman Group (sometimes referred to as the Oaklely Group) used to select initial Diffie-Hellman values. This must match the Diffie-Hellman Group configured on the remote system.

Acceptable Values:

1 (MODP, 768-bit exponent)
2 (1024-bit exponent)

Default: The default group parameter value is 2.

-hash MD5|SHA1

The hash argument specifies the hash algorithm for authenticating IKE messages. This must match the hash algorithm configured on the remote system.

Acceptable Values:

MD5 (128-bit key Hashed Message Authentication Code using RSA Message Digest-5, HMAC-MD5)
SHA1 (160-bit key HMAC using Secure Hash Algorithm-1, HMAC-SHA1)

Default: The default hash parameter value is MD5.

-encryption *encryption_algorithm*

>   The *encryption_algorithm* is the encryption algorithm for encrypting IKE messages. This must match the encryption algorithm configured on the remote system.
>
>   **Acceptable Values:**
>
>   **AES** (128-bit Advanced Encryption Standard, Cipher Block Chaining Mode, AES-CBC)
>   **DES** (56-bit Data Encryption Standard, Cipher Block Chaining Mode, DES-CBC)
>   **3DES** (triple-DES CBC, three encryption iterations, each with a different 56-bit key, 3DES-CBC)
>
>   **Default:** The default `encryption` parameter value is 3DES.

-life *lifetime_seconds*

>   The *lifetime_seconds* is the maximum lifetime for the IKE SA, in seconds.
>
>   **Range:** 0 (default), 30 - 4294967294 seconds (approximately 497102 days).
>
>   **Default:** 28,800 (8 hours).

Step 3: Configuring Preshared Keys Using Authentication Records

Complete this step only if you configured PSK (preshared keys) as an IKE authentication method in "Step 2: Configuring IKE Policies" (page 43). If you configured RSASIG (RSA signatures) as the IKE authentication method in all IKE policies, skip this step, and go to Section : "Step 4: Configuring Certificates" (page 52).

OpenVMS IPsec stores preshared keys in authentication records that are part of the Host Policy Information.

Configuring Preshared Keys without ID Information

Authentication records can also include IKE ID information. You do **not** have to configure IKE ID information if your topology meets the following requirements:

- You are using preshared keys.
- The remote system uses IP addresses as IKE IDs. OpenVMS IPsec systems use IP addresses as IKE IDs by default.
- You are using Main Mode (MM) for the IKE negotiations (you are not using Aggressive Mode). OpenVMS and most vendors use Main Mode by default.

If your topology does not meet the above requirements, you must configure IKE ID information.

`ipsec_config add auth` Syntax for Preshared Keys without ID Information

You can use the following `ipsec_config add auth` syntax to configure preshared keys without ID information in most installations:

`ipsec_config add auth auth_name`
`-remote ip_addr[/prefix] -preshared preshared_key`

HP recommends that you use an ipsec_config batch file to configure IPsec. To specify an add auth operation for an `ipsec_config` batch file, use the above syntax without the `ipsec_config` command name:

`add auth auth_name`
 `-remote ip_addr[/prefix] -preshared preshared_key`

The complete `ipsec_config add auth` syntax specification also allows you to specify the following arguments:

- `nocommit` (verify the syntax but do not commit the information to the database)
- `profile` (alternate profile file)
- `exchange` (exchange mode; if you do not configure ID information, you must use Main Mode, which is the default)
- `ltype` and `lid` (local ID type and value)
- `rtype` and `rid` (remote ID type and value)

auth_name

The `auth_name` is the user-defined name for the authentication record. This name must match the name given to the host policy with which the authentication record is associated.

Acceptable Values: 1 - 63 characters. Each character must be an ASCII alphanumeric character, hyphen (-), or underscore (_).

`-remote ip_addr [/prefix ]`

The ip_addr and `prefix` are the IP address and network prefix length that specifies the remote system or subnet for this record. Each `ip_addr` and `prefix` combination (the significant bits of `ip_addr`, as specified by `prefix`) must be unique.If the remote system's IP address matches multiple IP address and prefix combinations, IPsec uses the authentication record with the most specific address (longest prefix length).

Where:

ip_addr

The *ip_addr* is the remote IP address.

Acceptable Values: An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. The IP address type (IPv4 or IPv6) must be the same for the source and destination address. IPsec does not support unspecified IPv6 addresses. However, you can use the double-colon (::) notation within a specified IPv6 address to denote a number of zeros (0) within an address. The address cannot be a broadcast, subnet broadcast, multicast, or anycast address.

Default: None.

prefix

The *prefix* is the prefix length, or the number of leading bits that must match when comparing the remote IP address with *ip_addr*.

For IPv4 addresses, a prefix length of 32 bits indicates that all the bits in both addresses must match. This prefix length is equivalent to an address mask of 255.255.255.255. Use a value less than 32 to specify a subnet address filter.

For IPv6 addresses, a prefix length of 128 bits indicates that all the bits in both addresses must match. Use a value less than 128 to specify a subnet address filter.

⚠ **WARNING!** Specifying a subnet address filter and a preshared key allows you to configure a single preshared key for an entire subnet. However, HP strongly recommends that you configure an individual authentication record for each remote system with a unique preshared key.

Range: 0 - 32 for an IPv4 address; 0 - 128 for an IPv6 address. If you are using manual keys, prefix must be 32 if *ip_addr* is an IPv4 address or 128 if *ip_addr* is an IPv6 address.

Default: 32 if *ip_addr* is a non-zero IPv4 address, 128 if *ip_addr* is a non-zero IPv6 address, or 0 (match any address) if *ip_addr* is an all-zeros address (0.0.0.0 or 0::0).

-preshared *preshared_key*

The *preshared_key* is the preshared key used for IKE authentication. This must match the preshared key configured on the remote system.

Acceptable Values: A text string, containing 1 - 128 ASCII characters. White spaces are not allowed. You must quote shell special characters if you are using the command-line interface; do not quote them if you are using a batch file.

Default: None.

Configuring Preshared Keys with ID Information

You must configure IKE ID information with preshared keys for the following topologies:

- The remote system does not use IP addresses as IKE IDs. OpenVMS IPsec systems use IP addresses as IKE IDs by default.
- You are using Aggressive Mode (AM) for the IKE negotiations; you are not using Main Mode (MM).

As part of the IKE SA negotiation, the IKE peers exchange and verify ID types and ID values. For preshared key authentication, the authentication record contains the preshared key value and can also contain the following IKE ID information:

- local ID type
- local ID value

- remote ID type
- remote ID value

The OpenVMS IPsec Policy Manager retrieves authentication records as follows:

- If OpenVMS IPsec is the initiator in an IKEv1 Phase 1 negotiation (Main Mode or Aggressive Mode), IKE uses the remote system's IP address to search for an authentication record. If more than one authentication record matches the remote system's IP address, IKE uses the authentication record with the most specific (longest prefix length) IP address.
- If OpenVMS IPsec is a responder in an IKEv1 Phase 1 negotiation and the exchange type is Main Mode, IKE uses the remote system's IP address (from the IP packet header) to search for an authentication record.
- If OpenVMS IPsec is a responder in an IKEv1 Phase 1 negotiation and the exchange type is Aggressive Mode, IKE searches for an authentication record by comparing the ID information (the IKE Identity payload) sent by the remote system with the remote ID fields configured in the authentication records. IKE then uses the remote address field in the authentication record to search for the IKE policy.

If IKE uses an authentication record that specifies local ID information, OpenVMS IPsec sends the specified local ID in an IKE (ISAKMP) Identity payload. If IKE uses an authentication record that does not specify local ID information, OpenVMS IPsec sends the IP address of the interface it is using for the IKE negotiation as the local ID value, and sends the address type (IPv4 or IPv6) as the local ID type.

If IKE uses an authentication record that specifies remote ID information, OpenVMS IPsec uses the specified remote ID to verify what the remote system sends in the IKE (ISAKMP) Identity payload. If the matching authentication record does not specify remote ID information and the exchange mode is Main Mode, IPsec verifies that the source IP address from the inbound packet matches the ID value sent by the remote system, and uses the IP address type as the ID type.

ipsec_config add auth Syntax for Preshared Keys with ID Information

You can use the following `ipsec_config add auth` syntax to configure preshared keys in most installations:

```
ipsec_config add auth auth_name
 -remote ip_addr[/prefix] [-exchange|x AM|MM]
[-ltype local_id_type] [-lid local_id]
[-rtype remote_id_type] [-rid remote_id]
-preshared preshared_key
```

HP recommends that you use an ipsec_config batch file to configure IPsec. To specify an add auth operation for an `ipsec_config` batch file, use the above syntax without the `ipsec_config` command name:

```
add auth auth_name
 -remote ip_addr[/prefix] [-exchange|x AM|MM]
[-ltype local_id_type] [-lid local_id]
[-rtype remote_id_type] [-rid remote_id]
-preshared preshared_key
```

The complete `ipsec_config add auth` syntax specification also allows you to specify the following arguments:

- `nocommit` (verify the syntax but do not commit the information to the database)
- `profile` (alternate profile file)

auth_name

The *auth_name* is the user-defined name for the authentication record. This name must match the name given to the host policy with which the authentication record is associated.

Acceptable Values: 1 - 63 characters. Each character must be an ASCII alphanumeric character, hyphen (-), or underscore (_).

`-remote` `ip_addr` `[/prefix ]`

> The ip_addr and *prefix* are the IP address and network prefix length that specifies the remote system or subnet for this record. Each *ip_addr* and *prefix* combination (the significant bits of *ip_addr*, as specified by *prefix*) must be unique. If the remote system's IP address matches multiple IP address and prefix combinations, IPsec uses the authentication record with the most specific address (longest prefix length).

> Where:

> *ip_addr*

> The *ip_addr* is the remote IP address.

> **Acceptable Values:** An IPv4 address in dotted-decimal notation or an IPv6 address in colon-hexadecimal notation. The IP address type (IPv4 or IPv6) must be the same for the source and destination address. IPsec does not support unspecified IPv6 addresses. However, you can use the double-colon (::) notation within a specified IPv6 address to denote a number of zeros (0) within an address. The address cannot be a broadcast, subnet broadcast, multicast, or anycast address.

> **Default:** None.

> *prefix*

> The *prefix* is the prefix length, or the number of leading bits that must match when comparing the remote IP address with *ip_addr* .

> For IPv4 addresses, a prefix length of 32 bits indicates that all the bits in both addresses must match. This prefix length is equivalent to an address mask of 255.255.255.255. Use a value less than 32 to specify a subnet address filter.

> For IPv6 addresses, a prefix length of 128 bits indicates that all the bits in both addresses must match. Use a value less than 128 to specify a subnet address filter.

> ⚠ **WARNING!** Specifying a subnet address filter and a preshared key allows you to configure a single preshared key for an entire subnet. However, HP strongly recommends that you configure an individual authentication record for each remote system with a unique preshared key.

> **Range:** 0 - 32 for an IPv4 address; 0 - 128 for an IPv6 address. If you are using manual keys, prefix must be 32 if *ip_addr* is an IPv4 address or 128 if *ip_addr* is an IPv6 address.

> **Default:** 32 if *ip_addr* is a non-zero IPv4 address, 128 if *ip_addr* is a non-zero IPv6 address, or 0 (match any address) if *ip_addr* is an all-zeros address (0.0.0.0 or 0::0).

`-exchange AM|MM`

> Specifies the exchange mode for the IKEv1 Phase 1 negotiation. This must match what is configured on the remote system.

> **Acceptable Values:** AM (Aggressive Mode) or MM (Main Mode). Aggressive Mode does not provide identity protection (the IKE peers exchange identity information before establishing a secure channel), but it is more efficient.

> **Default:** MM (Main Mode).

> **TIP:** Most vendors use Main Mode by default. The IKE protocol specification requires implementations to support Main Mode; support for Aggressive Mode is optional.

`-ltype` *local_id_type* and `-lid` *local_id*

The *local_id_type* and *local_id* are the local ID type and value the local system sends to the remote system when negotiating an IKE SA. These values must match what is configured on the remote system.

You do not have to specify the local ID type and value if the local system uses IPv4 or IPv6 addresses as the ID type, and the local system is not multihomed. (OpenVMS IPsec uses IPv4 and IPv6 addresses for the ID type by default.)

Acceptable Values: Table 2-3 lists the valid ID types and corresponding ID values.

Table 2-3 ID Types and Values

ID Type	ID Value
`IPV4`	IPv4 address in dotted-decimal notation.
`IPV6`	IPv6 address in colon-hexadecimal notation.
`FQDN`	Fully Qualified Domain Name (FQDN), also known as Domain Name Server or DNS name, such as `myhost.hp.com`.
`KEY-ID`	Key identifier; a character string used to identify the preshared key. This is only valid for Aggressive Mode negotiations using preshared keys. You must also specify `-exchange` *AM* and `-preshared` *preshared_key*.
`USER-FQDN`	User-Fully Qualified Domain Name (User-FQDN) in SMTP format, such as `user@myhost.hp.com`.
`X500-DN`	X.500 Distinguished Name (DN). The format of the DN is described in the paragraphs that follow.

The DN consists of at least one of the following attributes:

```
CN=commonName
C=country
O=organization
OU=organizationalUnit
```

The attributes are all optional, but you must specify at least one. Separate multiple attributes using commas. The order of the attributes is ignored and the DN is not case sensitive.

If there are spaces in the DN, you must enclose the DN in double quotes (""). For example, `"CN=host1,C=US,O=My Company,OU=Blue Lab"`.

The values are defined as follows:

commonName: The commonName of the DN in printable string format. Commas are not accepted as part of this value. The size of this value must not exceed 64 bytes.

country: The two-character ISO 3166-1 code for the country listed in the DN, for example `US` for United States of America. Commas are not accepted as part of this value.

organization: The organization of the DN, for example `Hewlett-Packard`. Commas are not accepted as part of this value. The size of this value must not exceed 64 bytes.

organizationalUnit: The organizationalUnit for the DN, for example `Marketing`. Commas are not accepted as part of this value. The size of this value must not exceed 64 bytes.

Defaults: The address of the interface the local system uses to communicate with the remote system for the ID value and the appropriate IP address type (`IPV4` or `IPV6`) for the ID type.

-rtype *remote_id_type* and **-rid** *remote_id*

> The *remote_id_type* and *local_id* are used to verify the ID type and ID value sent by the remote system when negotiating a IKE SA. This must match what is configured on the remote system.
>
> You do not have to specify the remote ID type and value if the remote system is an OpenVMS system or a non-HP system that uses IPv4 or IPv6 addresses as the ID type, and is not multihomed.
>
> **Acceptable Values:** Table 2-3 (page 50) lists the valid ID types and corresponding ID values.
>
> **Defaults:** The *remote_id_type* and *remote_id* arguments are required if the IKE exchange mode is Aggressive Mode (-exchange AM). Otherwise, if *remote_id_type* and *remote_id* are not specified, OpenVMS uses the IP address of the remote system, from the source address of the inbound IP packets and the corresponding ID type (IPV4 or IPV6).

preshared_key

> The *preshared_key* is the preshared key used for IKE authentication. This must match the preshared key configured on the remote system.
>
> **Acceptable Values:** A text string, containing 1 - 128 ASCII characters. White spaces are not allowed. You must quote shell special characters if you are using the command-line interface; do not quote them if you are using a batch file.
>
> **Default:** None.

Step 4: Configuring Certificates

The EAK release does not currently support Security Certificates.

Step 5: Verifying the Batch File Syntax

Use the following command to verify the contents of the `ipsec_config` batch file without committing the configuration:

`ipsec_config batch batch_file_name -nocommit`

The ipsec_config utility displays the following message to indicate the profile file used:

```
Used default Profile file
TCPIP$IPSEC_DEVICE:[TCPIP$IPSEC]IPSEC_PROFILE.DAT
```

If there are no syntax errors in the batch file, `ipsec_config` returns without displaying any other messages.

Go on to "Step 6: Committing the Batch File Configuration and Verifying Operation" (page 54).

Step 6: Committing the Batch File Configuration and Verifying Operation

Use the following procedure to verify the operation of your OpenVMS IPsec configuration.

1. Commit the batch file operations to the configuration database with the following command:

   ipsec_config batch *batch_file_name*

2. The ipsec_config utility displays the contents of the configuration database. The contents include the configuration record automatically generated by ipsec_config for the default policy. The host policies are sorted in priority order. An output similar to the following will be displayed:

   ipsec_config show all

```
         host   apple_to_orange
       -source   192.6.2.50/32/0
  -destination   192.6.2.100/32/0
     -protocol   0
     -priority   99999999
       -action   ESP_AES128_HMAC_SHA1/28800/0

         host   default
       -action   PASS
```

3 Configuration Examples

Following are examples of configuring OpenVMS IPsec.

> **NOTE:** The host, IKE, and auth 'name's can be substituted with any name you choose. For
> example, in the command:
>
> ```
> ipsec_config add host policynamefoo
> ```
>
> *policynamefoo* may be substituted with a user selected name. Each 'name' in a given block of
> add commands for a single policy is required to match.

Example 3-1 HOST-TO-HOST Security Association (SA) using a Pre-Shared Key (PSK) AES128 encryption, SHA1 authentication

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -source 10.10.2.100
 -destination 10.10.2.200
 -action ESP_AES128_HMAC_SHA1

$ipsec_config add ike apple_to_orange
 -remote 10.10.2.200
 -authentication PSK

$ipsec_config add auth apple_to_orange
 -remote 10.10.2.200
 -preshared hello
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -source 10.10.2.200
 -destination 10.10.2.100
 -action ESP_AES128_HMAC_SHA1

$ipsec_config add ike orange_to_apple
 -remote 10.10.2.100
 -authentication PSK

$ipsec_config add auth orange_to_apple
 -remote 10.10.2.100
 -preshared hello
```

Example 3-2 HOST-TO-HOST SA using a PSK, AES128 encryp, MD5 auth, inbound/outbound TELNET

On node APPLE issue the following commands:

```
$ipsec_config add host telnet_apple_to_orange
 -source 10.10.2.100/32
 -destination 10.10.2.200/32/TELNET
 -action ESP_AES128_HMAC_MD5

$ipsec_config add ike telnet_apple_to_orange
 -remote 10.10.2.200/32
 -authentication PSK

$ipsec_config add auth telnet_apple_to_orange
 -remote 10.10.2.200
 -preshared hello
```

On node ORANGE issue the following commands:

```
$ipsec_config add host telnet_orange_to_apple
 -source 10.10.2.200
 -destination 10.10.2.100/32/TELNET
 -action ESP_AES128_HMAC_MD5

$ipsec_config add ike telnet_orange_to_apple
 -remote 10.10.2.100
 -authentication PSK

$ipsec_config add auth telnet_orange_to_apple
 -remote 10.10.2.100/32
 -preshared hello
```

Example 3-3 HOST-TO-HOST Security Association (SA) using a Pre-Shared Key (PSK) 3DES encryption, MD5 authentication, TCP traffic only, SA lifetime of 2 hours

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -source 10.10.2.100
 -destination 10.10.2.200
 -action ESP_3DES_HMAC_MD5/7200
 -protocol TCP

$ipsec_config add ike apple_to_orange
 -remote 10.10.2.200
 -authentication PSK

$ipsec_config add auth apple_to_orange
 -remote 10.10.2.200
 -preshared hello
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -source 10.10.2.200
 -destination 10.10.2.100
 -action ESP_3DES_HMAC_MD5/7200
 -protocol TCP

$ipsec_config add ike orange_to_apple
 -remote 10.10.2.100
 -authentication PSK

$ipsec_config add auth orange_to_apple
 -remote 10.10.2.100
 -preshared hello
```

Example 3-4 HOST-TO-HOST Security Association (SA) Tunnel Policy using a Pre-Shared Key (PSK), NULL encryption, MD5 authentication, IKE negotiations using AES encryption, IKE lifetime of 4 hours

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -source 10.10.2.100
 -destination 10.10.2.200
 -action ESP_NULL_HMAC_MD5
 -tunnel

$ipsec_config add ike apple_to_orange
 -remote 10.10.2.200
 -encryption AES
 -life 14400
 -authentication PSK

$ipsec_config add auth apple_to_orange
 -remote 10.10.2.200
 -preshared hello
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -source 10.10.2.200
 -destination 10.10.2.100
 -action ESP_NULL_HMAC_MD5
 -tunnel

$ipsec_config add ike orange_to_apple
 -remote 10.10.2.100
 -encryption AES
 -life 14400
 -authentication PSK

$ipsec_config add auth orange_to_apple
 -remote 10.10.2.100
 -preshared hello
```

Example 3-5 HOST-TO-HOST SA using manual keys, AES128 encryp, SHA1 auth

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -src 10.10.2.100
 -dst 10.10.2.200
 -action ESP_AES128_HMAC_SHA1
 -in
esp/2540010/0x12345678902234567890323456789042345678 90/0x1234567890abc
   def2234567890abcdef/0x4553502053412121
 -out
esp/2540011/0x12345678902234567890323456789042345678 90/0x1234567890ab
   cdef2234567890abcdef/0x4553502053412121
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -src 10.10.2.200
 -dst 10.10.2.100
 -action ESP_AES128_HMAC_SHA1
 -in
esp/2540011/0x12345678902234567890323456789042345678 90/0x1234567890abc
   def2234567890abcdef/0x4553502053412121
 -out
esp/2540010/0x12345678902234567890323456789042345678 90/0x1234567890ab
   cdef2234567890abcdef/0x4553502053412121
```

NOTE: The length of the encryption key for AES128 is 32 hex digits.

The length of the auth key for SHA1 is 40 hex digits.

Format is type/spi/auth_key[/encr_key[/iv]]

Example 3-6 HOST-TO-HOST SA using manual keys, 3DES encryp, SHA1 auth

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -src 10.10.2.100
 -dst 10.10.2.200
 -action ESP_3DES_HMAC_SHA1
 -in
esp/2540010/0x1234567890223456789032345678904234567890/0x1234567890ab
    cdef2234567890abcdef3234567890abcdef/0x4553502053412121
 -out
esp/2540011/0x1234567890223456789032345678904234567890/0x1234567890a
    bcdef2234567890abcdef3234567890abcdef/0x4553502053412121
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -src 10.10.2.200
 -dst 10.10.2.100
 -action ESP_3DES_HMAC_SHA1
 -in
esp/2540011/0x1234567890223456789032345678904234567890/0x1234567890ab
    cdef2234567890abcdef3234567890abcdef/0x4553502053412121
 -out
esp/2540010/0x1234567890223456789032345678904234567890/0x1234567890a
    bcdef2234567890abcdef3234567890abcdef/0x4553502053412121
```

📝 **NOTE:**   The length of the encryption key for 3DES is 48 hex digits.

The length of the encryption key for DES is 16 hex digits.

The length of the auth key for SHA1 is 40 hex digits.

Format is `type/spi/auth_key[/encr_key[/iv]]`.

Example 3-7 HOST-TO-HOST SA using manual keys, NULL (no) encryp, MD5 auth only

On node APPLE issue the following commands:

```
$ipsec_config add host apple_to_orange
 -src 16.116.92.100
 -dst 16.116.92.200
 -action ESP_NULL_HMAC_MD5
 -in esp/256/0x1234567890223456789032345678942
 -out esp/257/0x1234567890223456789032345678942
```

On node ORANGE issue the following commands:

```
$ipsec_config add host orange_to_apple
 -src 10.10.2.200
 -dst 10.10.2.100
 -action ESP_NULL_HMAC_MD5
 -in esp/257/0x1234567890223456789032345678942
 -out esp/256/0x1234567890223456789032345678942
```

NOTE: The length of the auth key for MD5 is 32 hex digits.

Format is `type/spi/auth_key`.

Example 3-8 Removing/Undoing `add` **Commands**

Use the following commands to remove (undo) `add`commands:

```
ipsec_config delete auth policynamefoo
ipsec_config delete ike policynamefoo
ipsec_config delete host policynamefoo
```

Index