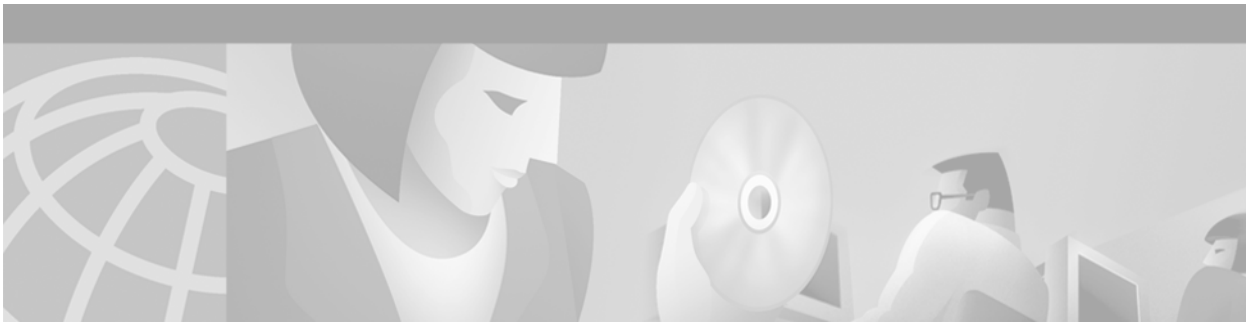


Configuring Internet Key Exchange Security Protocol

This chapter describes how to configure the Internet Key Exchange (IKE) protocol. IKE is a key management protocol standard that is used in conjunction with the IPSec standard. IPSec is an IP security feature that provides robust authentication and encryption of IP packets.

IPSec can be configured without IKE, but IKE enhances IPSec by providing additional features, flexibility, and ease of configuration for the IPSec standard.

IKE is a hybrid protocol that implements the Oakley key exchange and the Skeme key exchange inside the Internet Security Association and Key Management Protocol (ISAKMP) framework. (ISAKMP, Oakley, and Skeme are security protocols implemented by IKE.)

For a complete description of the IKE commands used in this chapter, refer to the "Internet Key Exchange Security Protocol Commands" chapter in the *Cisco IOS Security Command Reference*. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online.

To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the "Identifying Supported Platforms" section in the chapter "Using Cisco IOS Software."

In This Chapter

This chapter includes the following sections:

About IKE

IKE automatically negotiates IPSec security associations (SAs) and enables IPSec secure communications without costly manual preconfiguration. Specifically, IKE provides these benefits:

- Eliminates the need to manually specify all the IPSec security parameters in the crypto maps at both peers.
- Allows you to specify a lifetime for the IPSec security association.
- Allows encryption keys to change during IPSec sessions.
- Allows IPSec to provide anti-replay services.
- Permits certification authority (CA) support for a manageable, scalable IPSec implementation.
- Allows dynamic authentication of peers.

This section includes the following sections:

Supported Standards

Cisco implements the following standards:

- IKE—Internet Key Exchange. A hybrid protocol that implements Oakley and Skeme key exchanges inside the ISAKMP framework. IKE can be used with other protocols, but its initial implementation is with the IPSec protocol. IKE provides authentication of the IPSec peers, negotiates IPSec keys, and negotiates IPSec security associations.

  IKE is implemented in accordance with RFC 2409, *The Internet Key Exchange*.

- IPSec—IP Security Protocol. IPSec is a framework of open standards that provides data confidentiality, data integrity, and data authentication between participating peers. IPSec provides these security services at the IP layer; it uses IKE to handle negotiation of protocols and algorithms based on local policy and to generate the encryption and authentication keys to be used by IPSec. IPSec can be used to protect one or more data flows between a pair of hosts, between a pair of security gateways, or between a security gateway and a host.

  For more information on IPSec, see the chapter “Configuring IPSec Network Security.”

- ISAKMP—Internet Security Association and Key Management Protocol. A protocol framework that defines payload formats, the mechanics of implementing a key exchange protocol, and the negotiation of a security association.

  ISAKMP is implemented in accordance with the latest version of the *Internet Security Association and Key Management Protocol (ISAKMP)* Internet Draft (RFC 2408).

- Oakley—A key exchange protocol that defines how to derive authenticated keying material.
- Skeme—A key exchange protocol that defines how to derive authenticated keying material, with rapid key refreshment.

The component technologies implemented for use by IKE include the following:

- DES—Data Encryption Standard. An algorithim that is used to encrypt packet data. IKE implements the 56-bit DES-CBC with Explicit IV standard. Cipher Block Chaining (CBC) requires an initialization vector (IV) to start encryption. The IV is explicitly given in the IPSec packet.

  Cisco IOS software also implements Triple DES (168-bit) encryption, depending on the software versions available for a specific platform. Triple DES (3DES) is a strong form of encryption that allows sensitive information to be transmitted over untrusted networks. It enables customers, particularly in the finance industry, to utilize network-layer encryption.

  Note Cisco IOS images that have strong encryption (including, but not limited to, 56-bit data encryption feature sets) are subject to United States government export controls, and have a limited distribution. Images that are to be installed outside the United States require an export license. Customer orders might be denied or subject to delay because of United States government regulations. Contact your sales representative or distributor for more information, or send e-mail to export@cisco.com.

- Diffie-Hellman—A public-key cryptography protocol that allows two parties to establish a shared secret over an unsecure communications channel. Diffie-Hellman is used within IKE to establish session keys. 768-bit and 1024-bit Diffie-Hellman groups are supported.
- MD5 (HMAC variant)—Message Digest 5. A hash algorithm used to authenticate packet data. HMAC is a variant that provides an additional level of hashing.
- SHA (HMAC variant)—Secure Hash Algorithm. A hash algorithm used to authenticate packet data. HMAC is a variant that provides an additional level of hashing.
- RSA signatures and RSA encrypted nonces—RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Adleman. RSA signatures provide nonrepudiation, and RSA encrypted nonces provide repudiation. (Repudation and nonrepudation have to do with traceability.)

IKE interoperates with the following standard:

X.509v3 certificates—Used with the IKE protocol when authentication requires public keys. This certificate support allows the protected network to scale by providing the equivalent of a digital ID card to each device. When two devices wish to communicate, they exchange digital certificates to prove their identity (thus removing the need to manually exchange public keys with each peer or to manually specify a shared key at each peer).

List of Terms

Anti-Replay

Anti-replay is a security service in which the receiver can reject old or duplicate packets in order to protect itself against replay attacks. IPSec provides optional anti-replay services by use of a sequence number combined with the use of authentication.

Data Authentication

Data authentication includes two concepts:

- Data integrity (verifying that data has not been altered)
- Data origin authentication (verifying that the data was actually sent by the claimed sender)

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

Peer

In the context of this chapter, "peer" refers to a router or other device that participates in IPSec and IKE.

Perfect Forward Secrecy

Perfect forward secrecy (PFS) is a cryptographic characteristic associated with a derived shared secret value. With PFS, if one key is compromised, previous and subsequent keys are not also compromised, because subsequent keys are not derived from previous keys.

Repudiation

Repudation is a quality that prevents a third party from being able to prove that a communication between two other parties ever took place. This is a desirable quality if you do not want your communications to be traceable. **Nonrepudiation** is the opposite quality—a third party can prove that a communication between two other parties took place. Nonrepudiation is desirable if you want to be able to trace your communications and prove that they occurred.

Security Association

A security association (SA) describes how two or more entities will utilize security services to communicate securely. For example, an IPSec SA defines the encryption algorithm (if used), the authentication algorithm, and the shared session key to be used during the IPSec connection.

Both IPSec and IKE require and use SAs to identify the parameters of their connections. IKE can negotiate and establish its own SA. The IPSec SA is established either by IKE or by manual user configuration.

IKE Aggressive Mode Behavior

This section describes IKE aggressive mode behavior occurring when Cisco IOS software is used.

IKE has two phases of key negotiation: phase 1 and phase 2. Phase 1 negotiates a security association (a key) between two IKE peers. The key negotiated in phase 1 enables IKE peers to communicate securely in phase 2. During phase 2 negotiation, IKE establishes keys (security associations) for other applications, such as IPSec.

Phase 1 negotiation can occur using one of two modes: main mode and aggressive mode. Main mode tries to protect all information during the negotiation, meaning that no information is available to a potential attacker. When main mode is used, the identities of the two sides are hidden. Although this mode of operation is very secure, it is relatively costly in terms of the time it takes to complete the negotiation. Aggressive mode takes less time to negotiate keys between peers; however, it gives up some of the security provided by main mode negotiation. For example, the identities of the two parties trying to establish a security association are exposed to an eavesdropper.

The two modes serve different purposes and have different strengths. Main mode is slower than aggressive mode, but main mode is more secure and more flexible because it can offer an IKE peer more security proposals than aggressive mode. Aggressive mode is less flexible and not as secure, but much faster.

In Cisco IOS software, the two modes are not configurable. The default action for IKE authentication (rsa-sig, rsa-encr, or preshared) is to initiate main mode; however, in cases where there is no corresponding information to initiate authentication, and there is a preshared key associated with the host name of the peer, Cisco IOS software can initiate aggressive mode. Cisco IOS software will respond in aggressive mode to an IKE peer that initiates aggressive mode.

Whether Cisco IOS software initiates main mode or aggressive mode, the following restrictions are applicable:

- The initiating router *must not* have a certificate associated with the remote peer.
- The preshared key *must* be by fully qualified domain name (FQDN) on both peers.; thus, you have to enter the **crypto isakmp key** *keystring* **hostname** *peer-address* command in configuration mode.
- The communicating routers *must* have a FQDN host entry for each other in their configurations.
- The communicating routers *must* be configured to authenticate by hostname, *not* by IP address; thus, you should use the **crypto isakmp identity hostname** command.

IKE Configuration Task List

To configure IKE, perform the tasks in the following sections. The tasks in the first three sections are required; the remaining may be optional, depending on what parameters are configured.

- Enabling or Disabling IKE (Required)
- Ensuring That Access Lists Are Compatible with IKE (Required)
- Creating IKE Policies (Required)
- Manually Configuring RSA Keys (Optional, depending on IKE parameters)
- Configuring Preshared Keys (Optional, depending on IKE parameters)
- Configuring Mask Preshared Keys (Optional, depending on IKE parameters)
- Configuring Preshared Keys Using a AAA Server (Optional, depending on IKE parameters)
- Configuring Internet Key Exchange Mode Configuration (Optional)
- Configuring Internet Key Exchange Extended Authentication (Xauth) (Optional)
- Configuring Tunnel Endpoint Discovery (TED) (Optional)
- Clearing IKE Connections (Optional)
- Troubleshooting IKE (Optional)

For IKE configuration examples, refer to the section "IKE Configuration Examples" at the end of this chapter.

Enabling or Disabling IKE

IKE is enabled by default. IKE does not have to be enabled for individual interfaces, but it is enabled globally for all interfaces at the router.

If you do not want IKE to be used with your IPSec implementation, you can disable it at all IPSec peers.

If you disable IKE, you will have to make these concessions at the peers:

- You must manually specify all the IPSec security associations in the crypto maps at all peers. (Crypto map configuration is described in the chapter "Configuring IPSec Network Security.")
- The IPSec security associations of the peers will never time out for a given IPSec session.
- During IPSec sessions between the peers, the encryption keys will never change.
- Anti-replay services will not be available between the peers.
- Certification authority (CA) support cannot be used.

To disable or enable IKE, use one of the following commands in global configuration mode:

Command	Purpose
`Router(config)# no crypto isakmp enable`	Disables IKE.
`Router(config)# crypto isakmp enable`	Enables IKE.

If you disable IKE, you can skip the rest of the tasks in this chapter and go directly to IPSec configuration, as described in the chapter "Configuring IPSec Network Security."

Ensuring That Access Lists Are Compatible with IKE

IKE negotiation uses UDP on port 500. Ensure that your access lists are configured so that UDP port 500 traffic is not blocked at interfaces used by IKE and IPSec. In some cases you might need to add a statement to your access lists to explicitly permit UDP port 500 traffic.

Creating IKE Policies

You must create IKE policies at each peer. An IKE policy defines a combination of security parameters to be used during the IKE negotiation.

To create an IKE policy, follow the guidelines in these sections:

- Why Do You Need to Create These Policies?
- What Parameters Do You Define in a Policy?
- How Do IKE Peers Agree upon a Matching Policy?
- Which Value Should You Select for Each Parameter?
- Creating Policies
- Additional Configuration Required for IKE Policies

Why Do You Need to Create These Policies?

IKE negotiations must be protected, so each IKE negotiation begins by agreement of both peers on a common (shared) IKE policy. This policy states which security parameters will be used to protect subsequent IKE negotiations and mandates how the peers are authenticated.

After the two peers agree upon a policy, the security parameters of the policy are identified by a security association established at each peer, and these security associations apply to all subsequent IKE traffic during the negotiation.

You can create multiple, prioritized policies at each peer to ensure that at least one policy will match the policy of a remote peer.

What Parameters Do You Define in a Policy?

There are five parameters to define in each IKE policy:

Parameter	Accepted Values	Keyword	Default Value
encryption algorithm	56-bit DES-CBC 168-bit DES	**des** **3des**	56-bit DES-CBC 168-bit DES
hash algorithm	SHA-1 (HMAC variant) MD5 (HMAC variant)	**sha** **md5**	SHA-1
authentication method	RSA signatures RSA encrypted nonces preshared keys	**rsa-sig** **rsa-encr** **pre-share**	RSA signatures
Diffie-Hellman group identifier	768-bit Diffie-Hellman or 1024-bit Diffie-Hellman	**1** **2**	768-bit Diffie-Hellman
lifetime of the security association[1]	Any number of seconds	—	86400 seconds (one day)

1. For information about this lifetime and how it is used, see the command description for the **lifetime** command.

These parameters apply to the IKE negotiations when the IKE security association is established.

How Do IKE Peers Agree upon a Matching Policy?

When the IKE negotiation begins, IKE looks for an IKE policy that is the same on both peers. The peer that initiates the negotiation will send all its policies to the remote peer, and the remote peer will try to find a match. The remote peer looks for a match by comparing its own highest priority policy against the policies received from the other peer. The remote peer checks each of its policies in order of its priority (highest priority first) until a match is found.

A match is made when both policies from the two peers contain the same encryption, hash, authentication, and Diffie-Hellman parameter values, and when the remote peer's policy specifies a lifetime that is less than or equal to the lifetime in the policy being compared. (If the lifetimes are not identical, the shorter lifetime—from the remote peer's policy—will be used.)

If no acceptable match is found, IKE refuses negotiation and IPSec will not be established.

If a match is found, IKE will complete negotiation, and IPSec security associations will be created.

Note Depending on which authentication method is specified in a policy, additional configuration might be required (as described in the section "Additional Configuration Required for IKE Policies"). If a peer's policy does not have the required companion configuration, the peer will not submit the policy when attempting to find a matching policy with the remote peer.

Which Value Should You Select for Each Parameter?

You can select certain values for each parameter, in accordance with the IKE standard. But why chose one value over another?

If you are interoperating with a device that supports only one of the values for a parameter, your choice is limited to the value supported by the other device. Aside from this, there is often a trade-off between security and performance, and many of these parameter values represent such a trade-off. You should evaluate the level of security risks for your network and your tolerance for these risks. Then the following tips might help you select which value to specify for each parameter:

- The encryption algorithm has two options: 56-bit DES-CBC and 168-bit DES.
- The hash algorithm has two options: SHA-1 and MD5.

  MD5 has a smaller digest and is considered to be slightly faster than SHA-1. There has been a demonstrated successful (but extremely difficult) attack against MD5; however, the HMAC variant used by IKE prevents this attack.
- The authentication method has three options: RSA signatures, RSA encrypted nonces, and preshared keys.
  - RSA signatures provide nonrepudiation for the IKE negotiation (you can prove to a third party after the fact that you did indeed have an IKE negotiation with the remote peer).

    RSA signatures allow the use of a certification authority (CA). Using a CA can dramatically improve the manageability and scalability of your IPSec network. Additionally, RSA signature-based authentication uses only two public key operations, whereas RAS encryption uses four public key operations, making it costlier in terms of overall performance.

    You can also exchange the public keys manually, as described in section "Manually Configuring RSA Keys."
  - RSA encrypted nonces provide repudiation for the IKE negotiation (you cannot prove to a third party that you had an IKE negotiation with the remote peer).

    RSA encrypted nonces require that peers possess each other's public keys but do not use a certification authority. Instead, there are two ways for peers to get each other's public keys:

    1) During configuration you manually configure RSA keys (as described in the section "Manually Configuring RSA Keys").

    2) If your local peer has previously used RSA signatures with certificates during a successful IKE negotiation with a remote peer, your local peer already possesses the remote peer's public key. (The peers' public keys are exchanged during the RSA-signatures-based IKE negotiations, if certificates are used.)
  - Preshared keys are clumsy to use if your secured network is large, and they do not scale well with a growing network. However, they do not require use of a certification authority, as do RSA signatures, and might be easier to set up in a small network with fewer than 10 nodes. RSA signatures also can be considered more secure when compared with preshared key authentication.
- The Diffie-Hellman group identifier has two options: 768-bit and 1024-bit Diffie-Hellman.

  The 1024-bit Diffie-Hellman option is harder to crack, but requires more CPU time to execute.
- The lifetime of the security association can be set to any value.

  As a general rule, the shorter the lifetime (up to a point), the more secure your IKE negotiations will be. However, with longer lifetimes, future IPSec security associations can be set up more quickly. For more information about this parameter and how it is used, see the command description for the **lifetime** command.

Creating Policies

You can create multiple IKE policies, each with a different combination of parameter values. For each policy that you create, you assign a unique priority (1 through 10,000, with 1 being the highest priority).

You can configure multiple policies on each peer—but at least one of these policies must contain exactly the same encryption, hash, authentication, and Diffie-Hellman parameter values as one of the policies on the remote peer. (The lifetime parameter does not necessarily have to be the same; see details in the section "How Do IKE Peers Agree upon a Matching Policy?")

If you do not configure any policies, your router will use the default policy, which is always set to the lowest priority, and which contains the default value of each parameter.

To configure a policy, use the following commands, beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto isakmp policy priority`	Identifies the policy to create. (Each policy is uniquely identified by the priority number you assign.) (This command puts you into the config-isakmp command mode.)
Step 2	`Router(config-isakmp)# encryption {des | 3des}`	Specifies the encryption algorithm.
Step 3	`Router(config-isakmp)# hash {sha | md5}`	Specifies the hash algorithm.
Step 4	`Router(config-isakmp)# authentication {rsa-sig | rsa-encr | pre-share}`	Specifies the authentication method.
Step 5	`Router(config-isakmp)# group {1 | 2}`	Specifies the Diffie-Hellman group identifier.
Step 6	`Router(config-isakmp)# lifetime seconds`	Specifies the lifetime of the security association.
Step 7	`Router(config-isakmp)# exit`	Exits the config-isakmp command mode.
Step 8	`Router(config)# exit`	Exits the global configuration mode.
Step 9	`Router# show crypto isakmp policy`	(Optional) Displays all existing IKE policies. (Use this command in EXEC mode.)

If you do not specify a value for a parameter, the default value is assigned.

Note The default policy and the default values for configured policies do not show up in the configuration when you issue a **show running** command. Instead, to see the default policy and any default values within configured policies, use the **show crypto isakmp policy** command.

Additional Configuration Required for IKE Policies

Depending on which authentication method you specify in your IKE policies, you must do certain additional configuration tasks before IKE and IPSec can successfully use the IKE policies.

Each authentication method requires additional companion configuration as follows:

- RSA signatures method:

  If you specify RSA signatures as the authentication method in a policy, you may configure the peers to obtain certificates from a certification authority (CA). (The CA must be properly configured to issue the certificates.) Configure this certificate support as described in the chapter "Configuring Certification Authority Interoperability."

  The certificates are used by each peer to exchange public keys securely. (RSA signatures requires that each peer has the public signature key of the remote peer.) When both peers have valid certificates, they will automatically exchange public keys with each other as part of any IKE negotiation in which RSA signatures are used.

  You may also wish to exchange the public keys manually, as described in the section "Manually Configuring RSA Keys."

- RSA encrypted nonces method:

  If you specify RSA encrypted nonces as the authentication method in a policy, you must ensure that each peer has the public keys of the other peers.

  Unlike RSA signatures, the RSA encrypted nonces method can not use certificates to exchange public keys. Instead, you ensure that each peer has the others' public keys by one of the following methods:

  - Manually configuring RSA keys as described in the section "Manually Configuring RSA Keys."

    or

  - Ensuring that an IKE exchange using RSA signatures with certificates has already occurred between the peers. (The peers' public keys are exchanged during the RSA-signatures-based IKE negotiations if certificates are used.)

    To make this happen, specify two policies: a higher-priority policy with RSA encrypted nonces and a lower-priority policy with RSA signatures. When IKE negotiations occur, RSA signatures will be used the first time because the peers do not yet have each other's public keys. Then future IKE negotiations will be able to use RSA encrypted nonces because the public keys will have been exchanged.

    This alternative requires that you have certification authority support configured.

- Preshared keys authentication method:

  If you specify preshared keys as the authentication method in a policy, you must configure these preshared keys as described in the section "Configuring Preshared Keys."

If RSA encryption is configured and signature mode is negotiated (and certificates are used for signature mode), the peer will request both signature and encryption keys. Basically, the router will request as many keys as the configuration will support. If RSA encryption is not configured, it will just request a signature key.

Manually Configuring RSA Keys

Manually configure RSA keys when you specify RSA encrypted nonces as the authentication method in an IKE policy and you are not using a certification authority (CA).

To manually configure RSA keys, perform these tasks at each IPSec peer that uses RSA encrypted nonces in an IKE policy:

- Generating RSA Keys
- Setting ISAKMP Identity
- Specifying RSA Public Keys of All the Other Peers

Generating RSA Keys

To generate RSA keys, use the following commands starting in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto key generate rsa [usage-keys]`	Generates RSA keys.
Step 2	`Router# show crypto key mypubkey rsa`	Displays the generated RSA public key (in EXEC mode).

Remember to repeat these tasks at each peer (without CA support) that uses RSA encrypted nonces in an IKE policy.

Setting ISAKMP Identity

You should set the ISAKMP identity for each peer that uses preshared keys in an IKE policy.

When two peers use IKE to establish IPSec security associations, each peer sends its identity to the remote peer. Each peer sends either its host name or its IP address, depending on how you have set the ISAKMP identity of the router.

By default, a peer's ISAKMP identity is the IP address of the peer. If appropriate, you could change the identity to be the peer's host name instead. As a general rule, set the identities of all peers the same way—either all peers should use their IP addresses or all peers should use their host names. If some peers use their host names and some peers use their IP addresses to identify themselves to each other, IKE negotiations could fail if the identity of a remote peer is not recognized and a DNS lookup is unable to resolve the identity.

To set the ISAKMP identity of a peer, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto isakmp identity {address | hostname}`	At the local peer: Specifies the peer's ISAKMP identity by IP address or by host name.[1]
Step 2	`Router(config)# ip host hostname address1 [address2...address8]`	At all remote peers: If the local peer's ISAKMP identity was specified using a host name, maps the peer's host name to its IP address(es) at all the remote peers. (This step might be unnecessary if the host name or address is already mapped in a DNS server.)

1.See the **crypto isakmp identity** command description for guidelines for when to use the IP address and when to use the host name.

Remember to repeat these tasks at each peer that uses preshared keys in an IKE policy.

Specifying RSA Public Keys of All the Other Peers

At each peer, specify the RSA public keys of all the other peers by using the following commands starting in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto key pubkey-chain rsa`	Enters public key chain configuration mode.
Step 2	`Router(config-pubkey-c)# named-key` *key-name* [**encryption** | **signature**] or `Router (config-pubkey-c)# addressed-key` *key-address* [**encryption** | **signature**]	Indicates which remote peer's RSA public key you are going to specify. Enters public key configuration mode. If the remote peer uses its host name as its ISAKMP identity, use the **named-key** command and specify the remote peer's fully qualified domain name (such as somerouter.example.com) as the *key-name*. If the remote peer uses its IP address as its ISAKMP identity, use the **addressed-key** command and specify the remote peer's IP address as the *key-address*.
Step 3	`Router(config-pubkey-k)# address` *ip-address*	Specifies the remote peer's IP address. You can optionally use this command if you used a fully qualified domain name to name the remote peer in Step 2 (using the **named-key** command).
Step 4	`Router(config-pubkey-k)# key-string` *key-string*	Specifies the remote peer's RSA public key. This is the key previously viewed by the remote peer's administrator when the remote router's RSA keys were generated.
Step 5	`Router(config-pubkey-k)# quit`	Returns to public key chain configuration mode.
Step 6	—	Repeat Steps 2 through 4 to specify the RSA public keys of all the other IPSec peers that use RSA encrypted nonces in an IKE policy.
Step 7	`Router(config-pubkey-c)# exit`	Returns to global configuration mode.

Remember to repeat these tasks at each peer that uses RSA encrypted nonces in an IKE policy.

To view RSA public keys while or after you configure them, use the following command in EXEC mode:

Command	Purpose
`Router# show crypto key pubkey-chain rsa` {**name** *key-name* | **address** *key-address*}	Displays a list of all the RSA public keys stored on your router, or displays details of a particular RSA public key stored on your router.

Configuring Preshared Keys

To configure preshared keys, perform these tasks at each peer that uses preshared keys in an IKE policy:

- First, set the ISAKMP identity of each peer. Each peer's identity should be set to either its host name or by its IP address. By default, a peer's identity is set to its IP address. Setting ISAKMP identities is described in the section "Setting ISAKMP Identity."
- Next, specify the shared keys at each peer. Note that a given preshared key is shared between two peers. At a given peer you could specify the same key to share with multiple remote peers; however, a more secure approach is to specify different keys to share between different pairs of peers.

To specify preshared keys at a peer, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto isakmp key` *keystring* `address` *peer-address* or `Router(config)# crypto isakmp key` *keystring* `hostname` *peer-hostname*	At the local peer: Specifies the shared key to be used with a particular remote peer. If the remote peer specified its ISAKMP identity with an address, use the **address** keyword in this step; otherwise use the **hostname** keyword in this step.
Step 2	`Router(config)# crypto isakmp key` *keystring* `address` *peer-address* or `Router(config)# crypto isakmp key` *keystring* `hostname` *peer-hostname*	At the remote peer: Specifies the shared key to be used with the local peer. This is the same key you just specified at the local peer. If the local peer specified its ISAKMP identity with an address, use the **address** keyword in this step; otherwise use the **hostname** keyword in this step.
Step 3	—	Repeat Steps 1 and 2 for each remote peer.

Remember to repeat these tasks at each peer that uses preshared keys in an IKE policy.

Configuring Mask Preshared Keys

A mask preshared key allows a group of remote users with the same level of authentication to share an IKE preshared key. The preshared key of the remote peer must match the preshared key of the local peer for IKE authentication to occur.

A mask preshared key is usually distributed through a secure out-of-band channel. In a remote peer-to-local peer scenario, any remote peer with the IKE preshared key configured can establish IKE SAs with the local peer.

If you specify the *mask* argument with the **crypto isakmp key** command, it is up to you to use a subnet address, which will allow more peers to share the same key. That is, the preshared key is no longer restricted to use between two users.

Note Using 0.0.0.0 as a subnet address is not recommended because it encourages group preshared keys, which allow all peers to have the same group key, thereby reducing the security of your user authentication.

Mask preshared keys have the following restrictions:

- The SA cannot be established between the IPSec peers until all IPSec peers are configured for the same preshared key.
- The mask preshared key must be distinctly different for remote users requiring varying levels of authorization. You must configure a new preshared key for each level of trust and assign the correct keys to the correct parties. Otherwise, an untrusted party may obtain access to protected data.

Before configuring mask preshared keys, perform the tasks listed in the section "Configuring Preshared Keys."

To configure mask preshared key at each peer, use the following command in global configuration mode:

Command	Purpose
`Router(config)# crypto isakmp key` *keystring* `address` *peer-address* [*mask*]	At the local peer: Specifies the shared key to be used with a particular remote peer and the mask IP address. At the local peer: Specifies the shared key to be used with the local peer and the mask IP address. **Note** If you specify a mask, it is up to you to use a subnet address.

Configuring Preshared Keys Using a AAA Server

Preshared keys do not scale well when you are trying to deploy a large scale VPN without using a CA. When dynamic IP addressing such as DHCP or PPP dialups is used, the changing IP address can make key lookup difficult or impossible unless a mask preshared key is used. However, mask preshared keys are not very secure because a large number of users are given the same secret, thus reducing the security of the secret.

Configuring preshared keys using a AAA server allows each user to have his or her own key, which is stored on an external AAA server. This allows for central management of the user database, linking it to an existing AAA database, in addition to allowing every user to have a unique, more secure preshared key.

Preshared keys using a AAA server have the following restrictions:

- The shared secret can be accessed only in aggressive mode. The ID of the IKE exchange is used as the username to query AAA if no local key can be found on the Cisco IOS router to which the user is trying to connect. Aggressive mode provides the ID in the first part of the IKE exchange; main mode does not provide the ID until the latter part of the IKE exchange, which is too late for key lookup.
- Only the following ID types can be used:
  - IPV4 Address (can be different from the one assigned by the ISP)
  - FQDN (fully qualified domain name)
  - E-mail address

To configure this feature, perform the following tasks at each peer:

- Configure AAA.
- Configure an IPSec transform set.

- Configure a static crypto map.
- Configure extended authentication. (Optional)
- Configure ISAKMP policy.
- Configure a dynamic crypto map.

For information on configuring IPSec transform sets and crypto maps, refer to the chapter "Configuring IPSec Network Security."

To enable an IPSec peer for preshared keys using a AAA server, perform the following task in crypto map configuration mode:

Command	Purpose
`Router(config-crypto-map)# crypto map map-name isakmp authorization list list-name`	Enables IKE querying of AAA for tunnel attributes in aggressive mode.

Configuring Internet Key Exchange Mode Configuration

Internet Key Exchange (IKE) Mode Configuration, as defined by the Internet Engineering Task Force (IETF), allows a gateway to download an IP address (and other network level configuration) to the client as part of an IKE negotiation. Using this exchange, the gateway gives IP addresses to the IKE client to be used as an "inner" IP address encapsulated under IPSec. This method provides a known IP address for the client that can be matched against Internet Protocol Security (IPSec) policy.

To implement IPSec Virtual Private Networks (VPNs) between remote access clients that have dynamic IP addresses and a corporate gateway, you have to dynamically administer scalable IPSec policy on the gateway once each client is authenticated. With IKE Mode Configuration, the gateway can set up scalable policy for a very large set of clients irrespective of the IP addresses of those clients.

There are two types of IKE Mode Configuration:

- Gateway initiation—Gateway initiates the configuration mode with the client. Once the client responds, the IKE modifies the identity of the sender, the message is processed, and the client receives a response.
- Client initiation—Client initiates the configuration mode with the gateway. The gateway responds with an IP address that it has allocated for the client.

IKE Mode Configuration has the following restrictions:

- Interfaces with crypto maps that are configured for IKE Mode Configuration may experience a slightly longer connection setup time. This is true even for IKE peers that refuse to be configured or do not respond to the configuration mode request. In both cases, the gateway initiates the configuration of the client.
- This feature was not designed to enable the configuration mode for every IKE connection by default. Configure this feature at the global crypto map level.
- The following items in the IETF draft are not currently supported:
  - Configuration attributes other than INTERNAL_IP_ADDRESS
  - Unprotected exchanges

There are two steps to configuring IKE Mode Configuration on a router:

1. Define the pool of IP addresses.
2. Define which crypto maps should attempt to configure clients.

To configure IKE Mode Configuration on your Cisco access router, use the following commands in global configuration mode:

	Command	Purpose
Step 1	`router(config)# ip local pool pool-name start-addr end-addr`	Defines an existing local address pool that defines a set of addresses. For more information on the **ip local pool** command, refer to the *Cisco IOS Dial Technologies Command Reference*.
Step 2	`router(config)# crypto isakmp client configuration address-pool local pool-name`	References the local address pool in the IKE configuration. For more information on the **crypto isakmp client configuration address-pool local** command, refer to the *Cisco IOS Security Command Reference*.
Step 3	`router(config)# crypto map tag client configuration address [initiate | respond]`	Configures IKE Mode Configuration in global crypto map configuration mode. For more information on the **crypto map client configuration address** command, refer to the *Cisco IOS Security Command Reference*.

Configuring Internet Key Exchange Extended Authentication (Xauth)

IKE Extended Authentication (Xauth) is a draft RFC based on the IKE protocol. Xauth allows all Cisco IOS software AAA authentication methods to perform user authentication in a separate phase after the IKE authentication phase 1 exchange. The AAA configuration list name must match the Xauth configuration list name for user authentication to occur.

Xauth does not replace IKE. IKE allows for device authentication and Xauth allows for user authentication, which occurs after IKE device authentication. Xauth occurs after IKE authentication phase 1, but before IKE IPSec SA negotiation phase 2.

To configure Xauth, perform the following tasks:

- Configure AAA (You must set up an authentication list.)
- Configure an IPSec transform
- Configure a static crypto map
- Configure ISAKMP policy
- Configure a dynamic crypto map (Optional)

For information on configuring IPSec transform sets and crypto maps, refer to the chapter “Configuring IPSec Network Security.”

To enable Xauth on a crypto map, perform the following task in crypto map configuration mode:

Command	Purpose
`Router(config)# crypto map map-name client authentication list list-name`	Enables extended authentication (Xauth) on a crypto map. **Note** After enabling Xauth, you should apply the crypto map on which Xauth is configured to the router interface.

To verify that the Xauth feature is enabled, use the **show crypto map** command in EXEC mode. If the **crypto map client authentication list** command does not appear in the crypto map output, the Xauth feature is not enabled.

Configuring Tunnel Endpoint Discovery (TED)

Tunnel Endpoint Discovery (TED) is an enhancement to the IPSec feature. Defining a dynamic crypto map allows you to be able to dynamically determine an IPSec peer; however, only the receiving router has this ability. With TED, the initiating router can dynamically determine an IPSec peer for secure IPSec communications.

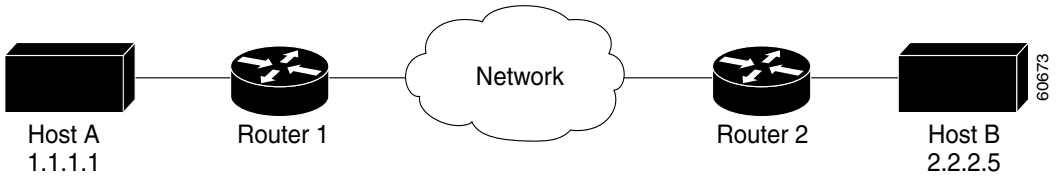
Dynamic TED helps to simplify IPSec configuration on the individual routers within a large network. Each node has a simple configuration that defines the local network that the router is protecting and the required IPSec transforms.

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

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

Figure 36 and the corresponding steps explain a sample TED network topology.

Figure 36 Tunnel Endpoint Discovery Sample Network Topology



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

Step 2 Router 1 intercepts and reads the packet. According to the IKE policy, Router 1 contains the following information: the packet must be encrypted, there are no SAs for the packet, and TED is enabled. Thus, Router 1 drops the packet and sends a TED probe into the network. (The TED probe contains the IP address of Host A (as the source IP address) and the IP address of Host B (as the destination IP address) embedded in the payload.)

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

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

Note IKE cannot occur until the peer is identified.

TED Versions

The following table lists the available TED versions:

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

TED Restrictions

Tunnel Endpoint Discovery has the following restrictions:

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

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

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

Note This restriction is no longer applicable in TEDv3.

To create a dynamic crypto map entry with Tunnel Endpoint Discovery (TED) configured, use the following commands, beginning in crypto-map configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto dynamic-map` *dynamic-map-name dynamic-map-number* `Router (config-crypto-m)# set transform-set` *transform-set-name1* [*transform-set-name2...transform-set-name6*] `Router (config-crypto-m)# match address` *access-list-id* `Router (config-crypto-m)# set security-association lifetime seconds` *seconds* and/or `Router (config-crypto-m)# set security-association lifetime kilobytes` *kilobytes* `Router (config-crypto-m)# set pfs` [group1 | group2] `Router (config-crypto-m)# exit`	Configures a dynamic crypto map using the **crypto dynamic-map** command. **Note** You *must* configure a **match address**; otherwise, the behavior is not secure, and you cannot enable TED because packets are sent in the clear (unencrypted.)
Step 2	`Router(config)# crypto map` *map-name map-number* `ipsec-isakmp dynamic` *dynamic-map-name* [**discover**]	Adds a dynamic crypto map to a crypto map set. Enter the **discover** keyword on the dynamic crypto map to enable TED.

Clearing IKE Connections

To clear IKE connections, use the following commands in EXEC mode:

	Command	Purpose
Step 1	`Router# show crypto isakmp sa`	Displays existing IKE connections; note the connection identifiers for connections you want to clear.
Step 2	`Router# clear crypto isakmp [connection-id]`	Clears IKE connections.

Troubleshooting IKE

To assist in troubleshooting IKE, use the following commands in EXEC mode:

Command	Purpose
`Router# show crypto isakmp policy`	Displays the parameters for each configured IKE policy.
`Router# show crypto isakmp sa`	Displays all current IKE security associations.
`Router# show crypto map`	Displays the crypto map configuration.
`Router# show running-config`	Verifies IKE configuration.
`Router# debug crypto isakmp`	Displays **debug** messages about IKE events.

What To Do Next

After IKE configuration is complete, you can configure IPSec. IPSec configuration is described in the chapter "Configuring IPSec Network Security."

IKE Configuration Examples

The following sections provide IKE configuration examples:

- Creating IKE Policies Examples
- Configuring Preshared Keys Using a AAA Server Example
- Configuring IKE Extended Authentication (Xauth) Examples

Creating IKE Policies Examples

This example creates two IKE policies, with policy 15 as the highest priority, policy 20 as the next priority, and the existing default priority as the lowest priority. It also creates a preshared key to be used with policy 20 with the remote peer whose IP address is 192.168.224.33.

```
crypto isakmp policy 15
 encryption 3des
 hash md5
 authentication rsa-sig
 group 2
 lifetime 5000
crypto isakmp policy 20
 authentication pre-share
 lifetime 10000
crypto isakmp key 1234567890 address 192.168.224.33
```

In the example, the **encryption des** of policy 15 would not appear in the written configuration because this is the default value for the encryption algorithm parameter.

If the **show crypto isakmp policy** command is issued with this configuration, the output is as follows:

```
Protection suite priority 15
encryption algorithm:3DES - Triple Data Encryption Standard (168 bit keys)
hash algorithm:Message Digest 5
authentication method:Rivest-Shamir-Adleman Signature
Diffie-Hellman group:#2 (1024 bit)
lifetime:5000 seconds, no volume limit
Protection suite priority 20
encryption algorithm:DES - Data Encryption Standard (56 bit keys)
hash algorithm:Secure Hash Standard
authentication method:preshared Key
Diffie-Hellman group:#1 (768 bit)
lifetime:10000 seconds, no volume limit
Default protection suite
encryption algorithm:DES - Data Encryption Standard (56 bit keys)
hash algorithm:Secure Hash Standard
authentication method:Rivest-Shamir-Adleman Signature
Diffie-Hellman group:#1 (768 bit)
lifetime:86400 seconds, no volume limit
```

Note that although the output shows “no volume limit” for the lifetimes, you can configure only a time lifetime (such as 86,400 seconds); volume-limit lifetimes are not configurable.

Configuring Preshared Keys Using a AAA Server Example

The following example shows how to configure a dynamic crypto map that will query a AAA server for a preshared key:

```
aaa new-model
aaa authorization network mylist group radius

!This defines the AAA server used for authorization.

crypto dynamic-map foo 10
    set security-association lifetime seconds 120
    set transform-set proposal1 proposal2
!
crypto map foo isakmp authorization list mylist
crypto map foo 10 ipsec-isakmp dynamic foo

! This sets up a dynamic crypto-map, which will query AAA for a shared secret.
```

Configuring IKE Extended Authentication (Xauth) Examples

The following sections provide examples of Xauth configurations with crypto maps:

- Configuring Xauth with Static Crypto Map Example
- Configuring Xauth with Dynamic Crypto Map Example

Configuring Xauth with Static Crypto Map Example

In the following example output from the **show running configuration** global configuration command, Xauth is configured with preshared key using AAA local policy:

```
aaa new-model
aaa authentication login xauthlist local
!
username robin password cisco1234
!
crypto ipsec transform-set xauthtransform esp-des esp-md5-hmac
!
crypto isakmp policy 1
 hash md5
 authentication pre-share
crypto isakmp key cisco1234 address 209.165.202.145
!
crypto map xauthmap client authentication list xauthlist
crypto map xauthmap 10 ipsec-isakmp
 set peer 209.165.202.145
 set transform-set xauthtransform
 match address 192
!
interface Ethernet1/0
 ip address 209.165.202.147 255.255.255.224
 crypto map xauthmap
!
access-list 192 permit ip host 209.165.202.147 host 209.165.202.145
```

Configuring Xauth with Dynamic Crypto Map Example

In the following example output from the **show running configuration** global configuration command, a corporate gateway uses Xauth configured on a RADIUS authentication server. Digital certification is also configured with dynamic crypto maps for scalability. This allows for both remote user authentication and device authentication.

```
aaa new-model
radius-server host alcatraz
radius-server key cisco12345
aaa authentication login xauthlist radius
!
crypto ipsec transform-set remote esp-des esp-md5-hmac
!
crypto ca identity myca
    enrollment url http://myca.cisco.com:80
crypto ca certificate chain myca
    certificate ca <cert-serial-number>
    <hex data>
    certificate
    <hex data>
!
```

```
crypto dynamic-map xauthdynamic 10
    set transform-set xauthtransform
!
crypto map xauthmap client authentication list xauthlist
crypto map xauthmap 10 ipsec-isakmp dynamic xauthdynamic
!
interface Ethernet1/0
    ip address 209.165.202.147 255.255.255.224
    crypto map xauthmap
```