



# IPSec and Virtual Private Networks

---

## **Terms you'll need to understand:**

- ✓ ISAKMP policies
- ✓ Internet Key Exchange (IKE)
- ✓ Internet Protocol Security (IPSec)
- ✓ Authentication header (AH)
- ✓ Encapsulation security payload (ESP)
- ✓ Transform sets
- ✓ Crypto maps
- ✓ IP local pool
- ✓ Security association

## **Techniques you'll need to master:**

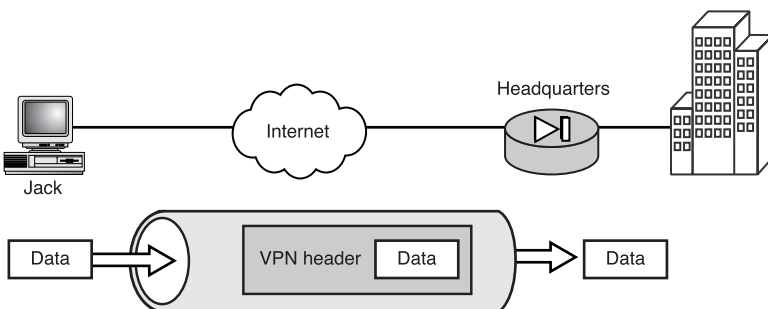
- ✓ Four steps to setting up IPSec
- ✓ ISAKMP policies
- ✓ Transform sets
- ✓ Crypto maps

Before the innovation of virtual private networks (VPNs), companies had no choice but to purchase expensive wide area network (WAN) connections to interlink sites or allow users access into their networks. These days, companies and home users alike can use VPNs to access network resources—not by traveling across dedicated WAN links, but by using public media such as the Internet. This chapter contains a high-level overview of VPN technology and discusses VPN functionality provided by the PIX firewall.

## The Basics of VPN

The basic concept of a VPN is really quite simple: If a user wants to send traffic from one point to another, that data is placed inside another packet and sent to its destination. This process is known as *encapsulation*. For example, if Jack wants to send traffic from his computer to the headquarters office across the Internet, Jack first establishes a secure VPN tunnel with HQ. Then, all the traffic directed to HQ is broken up and placed into other packets being encapsulated, and perhaps even encrypted from prying eyes.

To give an analogy, VPNs are like having truck carry a payload of data for you. If you want to send data to another location, you place that data in the truck. Instead of an open, flatbed truck where everyone can see your data, the truck has a canopy enclosing (encrypting) the payload so other users cannot view your true data. Figure 12.1 displays Jack's traffic flowing through a remote access VPN tunnel to HQ. After the traffic reaches HQ, the data is taken out of the tunnel, reassembled, and sent along its way. As you can see, VPNs just perform a middleman action to carry data from one peer to another in a secure manner.

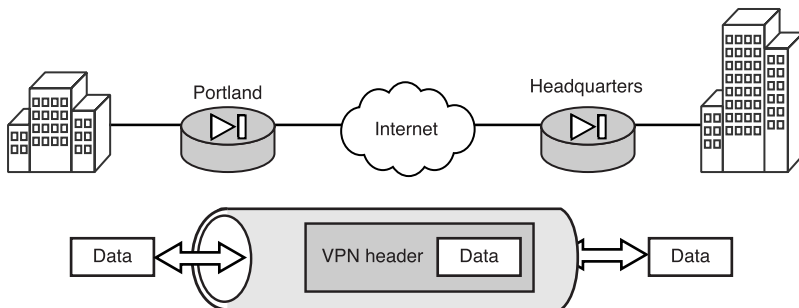


**Figure 12.1** Basic access VPN traffic.

## VPN Categories

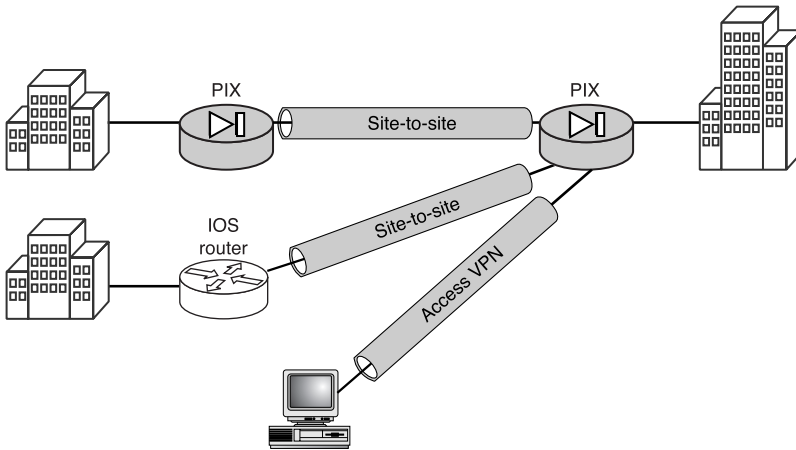
VPNs can be divided into several categories to help define what type of connection you've created. The following is a list of the most common categories:

- *Access VPN*—These VPNs provide a means for remote users to access the network in a secure manner. For example, say Jack wants to connect from home temporarily to the office to check his email. Jack can set up a VPN to interlink his home PC to the office and send data securely. This is called an access VPN (refer to Figure 12.1)
- *Intranet VPN*—This allows a company to interconnect its remote networks. Intranet VPNs are commonly known as *site-to-site* VPNs and are used to link sites that are part of the same company. They don't just link a single user but whole networks. Figure 12.2 demonstrates a VPN across a site-to-site scenario.
- *Extranet VPN*—This is similar to the intranet VPN. However, this VPN solution is used when interlinking two different companies. For example, if company A wants to interconnect with company B, it can use the Internet as a backbone and set up a site-to-site VPN called an extranet VPN.



**Figure 12.2** A site-to-site VPN.

The PIX firewall can perform all the functions of the previously mentioned categories, including linking site-to-site VPNs and remote user access VPNs. The PIX is capable of interlinking with other PIX firewalls, clients, routers, and even third-party firewalls, to name a few. Figure 12.3 shows various combinations using the PIX firewall.



**Figure 12.3** Using the PIX for VPNs.

## Types of VPNs

The use of the term *VPN* makes it sound like a single thing, when actually it's just a technology with several types and flavors from which to choose. Several VPN types are available, but we will cover only the three main types used with the PIX: PPTP, L2TP, and IPsec.

### PPTP

The Point-to-Point Tunneling Protocol (PPTP) VPN solution, primarily created by Microsoft, makes Point-to-Point (PPP) traffic routable. It provides for user authentication and uses Microsoft Point-to-Point Encryption (MPPE) to secure traffic. The PIX firewall supports PPTP for remote access; however, PPTP isn't supported by all vendors.

### L2TP

Layer 2 Tunneling Protocol (L2TP) VPNs are an enhancement of the Cisco Layer 2 Forwarder (L2F) mechanism that works only at layer 2 to forward IP, IPX, and AppleTalk traffic. L2TP builds on L2F to make it routable across IP networks. This work was done by a combination of efforts from Cisco, Microsoft, Ascent, and 3Com to form RFC 2661. The L2TP VPN solution doesn't contain any encryption engine built in the way PPTP does. However, L2TP is typically implemented with IPsec to create what is called L2TP over IPsec, making L2TP very secure with the added benefit of user

authentication. This tunneling protocol can authenticate on both user and machine levels, giving you more granularity over who can connect. Similar to PPTP, L2TP is typically used for remote access.

## IPSec

IPSec VPN is an open standard defining a group of security protocols used together to form a secure connection between two peers. Basically, IPSec is a VPN tunnel that enables you to encrypt traffic or guarantee that it hasn't changed from one peer to another. The PIX firewall supports site-to-site and access VPN traffic with IPSec. Most products on the market that support VPNs also support IPSec for interoperability with other vendors.

Of the three types of VPNs covered here, IPSec is the main focus of this chapter and the following sections.

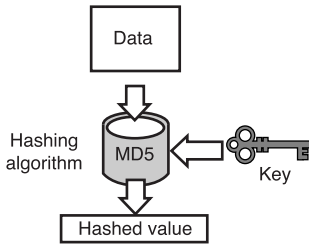
# Defining Hashing, Encryption, and Keys

Before we dive into the deep dark depths of how IPSec VPNs work, let's take a moment to review hashing, encryption, and the keys used for everything throughout.

## Hashing

*Hashing* is not encryption, but actually a result from an algorithm. When data, a key, and an algorithm are combined, a fixed result is generated. This result is a small, fixed-length piece of data, which you could call a fingerprint (message digest or hash) of the data and key. Every time the exact data and key are run through the algorithm, the exact same output value is produced. A modification of even 1 bit in the data or key produces an entirely different fingerprint. For example, if Jack wants to guarantee that his data doesn't change when he sends it to Peter, he could hash the data. After he has the hash result known as a *digital signature*, he can send this along with the data. When Peter receives the data and digital signature, he can use the same data, key, and hashing algorithm to generate a result himself. If this hash result is exactly the same as the signature Jack sent, the data is true and hasn't been modified.

Now for all of this to work, three things are needed: data, a key, and an algorithm. Both sides need all three to come up with the same result. Two algorithms commonly used in IPSec are Message Digest version 5 and Secure Hash Algorithm. Figure 12.4 displays an example of hashing data to get a single hashed value.



**Figure 12.4** Hashing example.

## Message Digest Version 5

Message Digest version 5 (MD5) is a hashing algorithm commonly used to authenticate data and ensure that data hasn't changed. It is a one-way hash that produces a fixed-length result called a *message digest*. MD5 feeds successive 512-bit blocks of data into the algorithm to eventually produce a 128-bit message digest.

## Secure Hash Algorithm

Secure Hash Algorithm (SHA-1) is the second hashing algorithm that can be used by IPSec. It produces a 160-bit result and is considered more secure than MD5. Because SHA-1 is more secure, it also takes longer to perform its functions.

## Encryption

*Encryption* is the process of taking data, a key, and an encryption algorithm and producing encrypted data known as *cipher text*. If the encrypted data needs to be extracted back to normal, a decryption key and algorithm are necessary. The following sections discuss the two main types of encryption used by IPSec: DES and 3DES.

## Data Encryption Standard

Data Encryption Standard (DES) uses a 56-bit symmetric encryption key, meaning the 56-bit key used to encrypt the data is the same key used to decrypt the data. DES was published in 1977 and is considered a fast and

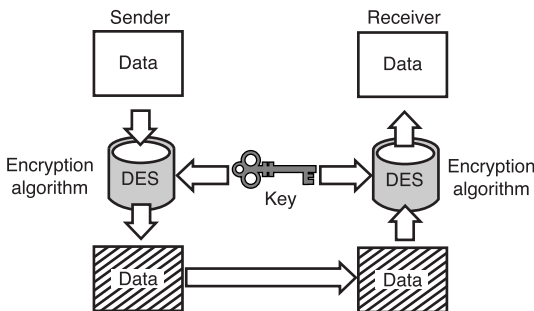
reliable encryption engine that is commonly used to provide basic protection or in places outside the United States where the export of stronger encryption is not allowed.



In 1998, the Electronic Frontier Foundation (EFF) built the first unclassified DES encryption cracker. This system took only three days to crack DES, proving it to be an insecure algorithm. Six months later, the Distributed.Net, a worldwide coalition of computer enthusiasts, networked 100,000 computers together to crack DES in only 22 hours. Needless to say, DES is considered to be unsuitable for highly secure environments. As of this writing, triple DES has not been cracked.

## Triple DES

Triple DES (3DES) is a spin-off of DES itself. Basically, 3DES encrypts data, but it does it three times. 3DES iterates through the data three times with three different keys, dramatically increasing the protection level of the data. 3DES uses a 168-bit key and takes more time and processor power to run than normal DES does. Figure 12.5 shows an example of data encryption.



**Figure 12.5** An encryption example.



The Advanced Encryption Standard (AES) is a recent encryption algorithm standard sponsored by the National Institute of Standards and Technology. AES is gaining acceptance and is supported in most newer IPSec implementations in the place of 3DES. AES processing is faster and supports larger key sizes than 3DES does, making it a better choice based on speed and level of protection provided. This book mainly covers DES and 3DES because support for the older algorithms is widely used. See <http://www.nwfusion.com/links/Encyclopedia/A/597.html> for more information about the newer AES algorithm.

## Keys

The two technologies, hashing and encryption, both require keys with their algorithms. Keys come in two main types—symmetric and asymmetric. You will see how these keys are used in several ways with different algorithms and technologies, but they all follow the same basic concepts.

## Symmetric Keys

The *symmetric* key, also known as a *shared secret* key, is a single key used by both parties involved. This key is typically used to encrypt and decrypt data. For example, if Jack uses a key of  $\times 132w$  to encrypt his data, Peter will need to use the same  $\times 132w$  to decrypt that data. The hardest thing about using symmetric keys is getting the same key to the other side without anyone ever knowing what it is. Lastly, symmetric keys are generally used by very fast systems and are used to encrypt and decrypt bulk amounts of data.

## Asymmetric Keys

The *asymmetric* key is actually two keys paired together. One is called the *public* key, and the other is called the *private* key. The public key is given out freely, whereas the private key never leaves the owner. When you encrypt something with the public key, it takes the corresponding private key to decrypt it, or vice versa. For example, if Jack has a copy of Peter's public key and uses that public key to encrypt some data, Peter could use his own matching private key to decrypt that data. Therefore, only Peter could decrypt the data. Asymmetric keys are considered slower and take more processor power; however, they are great for authentication and for use during the authentication phases of communication.

# Internet Protocol Security

Internet Protocol Security (IPSec) is not really a protocol, as the name would suggest—it is actually a framework of many protocols and mechanisms working together to produce a secure connection between two peers. The heart of IPSec works at the Network layer of the OSI model. Some of the features this framework provides are as follows:

- ▶ *Data integrity*—A mechanism used to guarantee that data has not changed. The MD5 or SHA-1 hashing algorithm is used to produce a message digest of the data that is then cross-checked at the destination for data integrity. IPSec uses the MD5 and SHA-1 hashing algorithms to perform this function.
- ▶ *Data origin authentication*—Authenticates the source of the data against the peer. For this to work, a data integrity feature is necessary.
- ▶ *Data confidentiality*—The technical form of saying the data is encrypted, meaning that confidential information is encrypted so no one else can make sense of it. IPSec uses symmetric key encryption algorithms (DES, 3DES, AES) for data confidentiality.



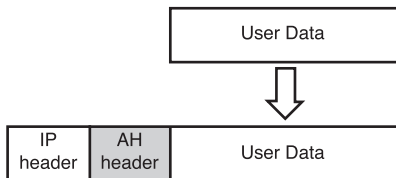
- *Anti-replay*—A feature that helps protect against hackers who want to cause harm by sending the same packet data repeatedly. Anti-replay helps guarantee that data is received only once.

## IPSec Components

IPSec contains several separate components that all work together to make a secure connection between two peers. This section highlights these components; then we will link them all together to make an IPSec Security Association (SA).

### Authentication Headers

The authentication header (AH) mode of IPSec provides data authentication and anti-replay protection. Data authentication is also known as data *integrity*, meaning that AH checks to ensure that the data has not been altered. AH also provides data origin authentication to ensure that the data is coming from the correct source. AH doesn't actually encrypt data. Also, if it's used alone, it does not provide data confidentiality. Figure 12.6 shows user data being placed in an AH-protected packet.



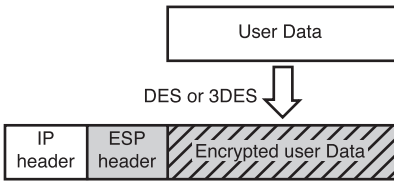
**Figure 12.6** An AH example.

### Encapsulation Secure Payload

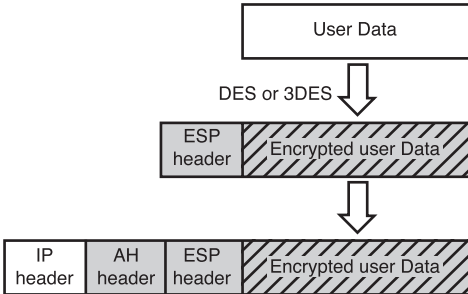
Encapsulation Secure Payload (ESP) performs what most people need, data confidentiality. ESP also performs data authentication and anti-replay protection. ESP encrypts the whole payload, which contains layer 4 headers and the data. ESP uses protocol port 50 and can either be used with AH or stand alone to provide data confidentiality. Figure 12.7 shows user data being encrypted and protected within ESP.



AH, which uses protocol port 51, helps protect against session hijacking, whereas ESP uses protocol port 50 and encrypts the data. Both AH and ESP can be used separately or together. When they are both used, ESP encryption is processed before the AH digest is created. Figure 12.8 shows data being encrypted by ESP and encapsulated by AH.



**Figure 12.7** An ESP example.



**Figure 12.8** ESP encapsulated in an AH.

## Internet Key Exchange

Internet Key Exchange (IKE) is a hybrid of several other protocols such as ISAKMP and the Oakley and Skeme key exchange. When two computers (peers) connect, they need to exchange policies and keys for hashing and encryption. IKE and ISAKMP perform these functions.



IKE is a hybrid protocol to exchange keys and uses UDP port number 500.

## Diffie-Hellman

The Diffie-Hellman (D-H) algorithm is actually part of the IKE process to exchange keys safely and securely. D-H uses a pair of asymmetric keys, and each peer's public key is exchanged with the other peer's. These public keys are then combined with the others' private keys to generate an identical symmetric key on both peers. This new shared secret (symmetric) key is used to provide encryption and decryption during the IKE establishment phases. The PIX is capable of using either 768-bit (group 1) or 1024-bit (group 2) D-H groups.

## Authentication

When two IPSec peers connect, they need to authenticate before secure data can be transmitted. This authentication can be done in several ways; the following list describes three methods that can be used to authenticate peers:

- ▶ *Pre-shared keys*—These are hard-coded values that are set on both peers. During authentication, if both values match, authentication succeeds. For example, if Jack codes the word “dog” and Peter also uses the word “dog,” this would provide a positive authentication. This is by far the easiest to set up; however, if you had 100 VPN peers and wanted to change the key of “dog” to “cat,” you have a lot of work to do.
- ▶ *RSA encrypted nonces*—This is a time-variant mechanism that uses asymmetric public and private keys that need to be manually created on each peer. Then the public keys must be shared with the peers that will be connecting. When VPN peers connect, each peer uses its own private key to create a digital signature, which is then sent across for authentication. Each peer uses the corresponding shared public key to verify the digital signature. RSA nonces are time-consuming to set up, and if they need to be changed, they take even more time and effort to change than pre-shared keys.
- ▶ *RSA signatures*—These are very similar to nonces and use asymmetric public and private key pairs. However, the use of a certificate authority (CA) is involved with certificate generation and authentication. During IKE phase 1, peers exchange each other’s certificates. Then, they contact the respective CA to validate the received peer’s certificate. This process enables you to change a peer’s certificate; and all connecting peers don’t need to be modified because the new certificate is sent down during the next connection. RSA signatures is a very scalable mechanism when supporting thousands of VPN peers.



Diffie-Hellman is susceptible to man-in-the-middle attacks. To mitigate this problem, authentication is used during the D-H key agreement algorithm. The authentication methods used are pre-shared keys, nonces, and RSA signatures.

## Security Association

The security association (SA) is similar to a session between TCP hosts. Whenever two peers make a successful transfer of data using IPSec, an SA is created in the background to maintain the connection. This SA identifies each peer by IP address, security protocols, and a security parameter index (SPI).



Security associations can be created using IKE or a manual process. See Cisco for more details about manual security associations.

## How IPSec Works

Now that we've reviewed hashing, keys, encryption, AH, and ESP, let's put them all together and see how the IPSec framework makes a secure connection between two peers. IPSec is similar to creating a TCP connection that uses a three-way handshake. The peers exchange several parameters to create a security association; then, after the security association has been established, data can be sent in a protected manner. This exchange is comprised of two main phases called phase 1 and phase 2.

Following are some of the steps needed to create an IPSec security association between two peers:

1. Connect to a peer.
2. IKE phase 1 starts, which involves the following:
  - 2a. IKE exchange polices
  - 2b. Diffie-Hellman key agreements are set
  - 2c. Authentication
3. IKE phase 2 starts (this is the IPSec phase), causing the following actions:
  - 3a. Crypto maps are exchanged.
  - 3b. The SA lifetime is established.
4. When data that matches the crypto ACL is sent, it is protected.

To prepare you, phase 1 and phase 2 are where most of your parameters need to be configured between two peers. If they match, a successful security association can be made.

### Phase 1

IKE controls phase 1 to create a management connection between the two peers. The management connection exchanges ISAKMP policies, keys, and authentication information. ISAKMP polices are negotiated parameter sets that each peer uses to communicate. Some of the parameters inside an ISAKMP policy include

- The encryption algorithm used is DES or 3DES.
- The hashing algorithm used is MD5 or SHA-1.
- The Diffie-Hellman group used is group 1 or group 2.
- The authentication used is pre-shared keys, RSA nonces, or RSA signature with CAs.
- The lifetime of the management connection.

Peers can contain multiple policies; as peers connect, they iterate through the policies until a matching set is found on both sides. For example, say Jack's policy uses DES and MD5 and Peter's policy uses 3DES and MD5. Because these policies don't match, Jack and Peter will not be able to connect. However, if Peter's policy contains a second policy that matches Jack's, this phase will succeed and they can move on to the next step.

Phase 1 also uses Diffie-Hellman to generate symmetric keys for encryption or hashing. Finally, the authentication is completed between the two peers. When all this is completed, enough information exists to make a secure connection to send IPSec parameters in phase 2.




Phase 1 has two modes of operation: main and aggressive. Main mode takes three exchanges to complete and hides the peer's identity. Aggressive mode takes only two exchanges, making it faster, or more aggressive as it were. However, aggressive mode doesn't hide the peer's identity.

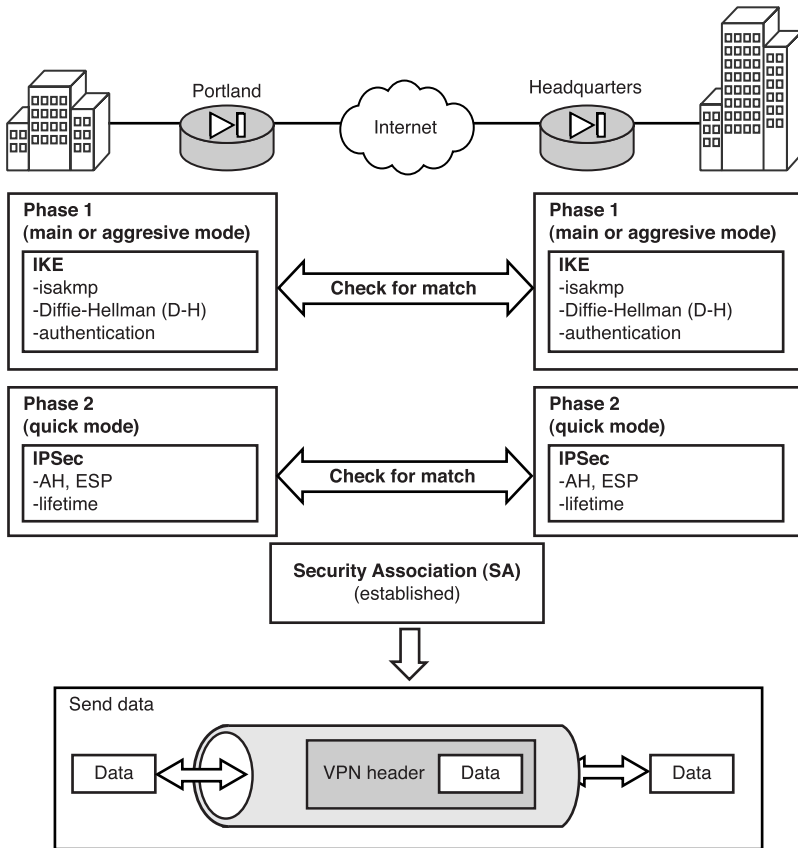
## Phase 2

After phase 1 is completed, IKE starts phase 2. Phase 2 creates a user connection that negotiates IPSec parameters such as the transform sets. Transform sets are the modes or methods the two peers use to protect user data—for instance, AH, ESP, or both. Phase 2 also negotiates the encryption key, hash keys, and lifetime of how long the keys are valid before regeneration is necessary. When all these processes are complete, an SA is formed. The parameters contained on a crypto map that are sent between peers are as follows:

- The security protocol is AH, ESP, or both.
- The encryption algorithm for ESP is DES or 3DES.
- The authentication method is AH, ESP, or both.
- The authentication hashing algorithm is MD5 or SHA-1.
- The ESP mode is tunnel or transport.
- The lifetime is time in seconds, amount of data transmitted, or both.

After phase 2 is complete, peers start to send data securely. Figure 12.9 displays phase 1 and phase 2 checking for matching parameters before creating a security association and sending data.

 Phase 2 has only one mode, which is called quick mode. Because it's already using a secure connection, phase 2 only needs to send parameters.



**Figure 12.9** Phase 1 and phase 2 example.

# Configuring an IPSec Site-to-Site Connection

Next, let's discuss configuring IPSec on a PIX firewall. This example shows you how to configure a site-to-site VPN using pre-shared keys. The four main tasks you perform when setting up the VPN connection are

- *Preparing for IPSec*—Involves gathering IPSec parameter details.
- *Configure IKE*—Phase 1 parameters are configured.
- *Configure IPSec*—Phase 2 parameters are configured.
- *Testing and troubleshooting*—This is when configuration and current security associations are displayed.

The following sections discuss each task in detail.

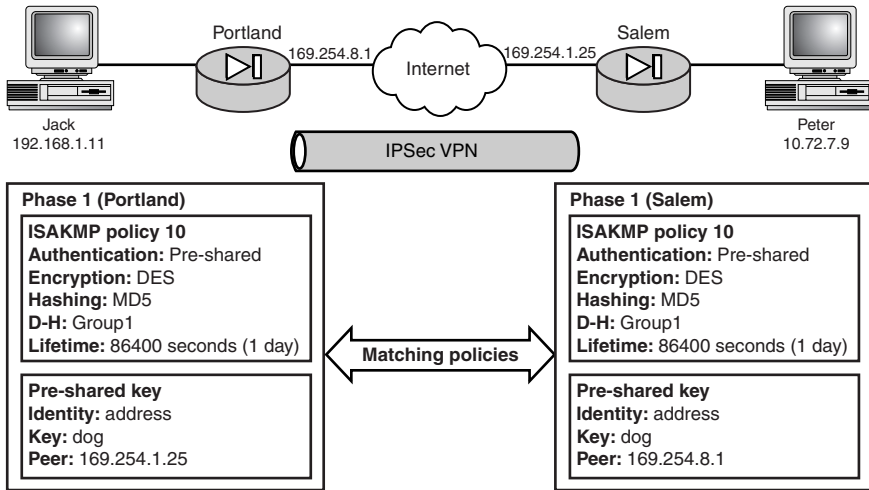
## Preparing for IPSec

In any implementation of IPSec, the preparation phase is a key element of a successful installation. This task enables you to document and plan all the settings that will be needed to configure IPSec. If any policies or transform sets don't match, your IPSec connection can fail. The steps needed to complete this task include the following:

1. Define the IKE phase 1 policies between peers.
2. Define the IKE phase 2 policies, transform sets, peer IP address or hostnames, and lifetime settings.
3. Verify the current PIX configuration for any previous access control list (ACL), ISAKMP policies, or crypto maps that might conflict with the new settings.
4. Perform a basic ping test to ensure you can actually reach the other peer before attempting to create a VPN.
5. Verify that perimeter routers will allow IPSec traffic using protocol 50, 51, and IKE UDP port 500 to pass through.

Your goal is to configure a site-to-site VPN from the Portland firewall to the Salem firewall. All traffic from the internal Portland LAN 192.168.1.0/24 will be protected by the VPN tunnel only if it's traveling to the Salem LAN

of 10.0.0.0/8. Figure 12.10 shows an overview of the phase 1 settings. Note that these are just the parameters needed in phase 1 and loosely related to the actual commands you will use later to configure this phase.



**Figure 12.10** Phase 1 settings.

Figure 12.11 shows the phase 2 information for creating a crypto map called MapPtoS. The crypto map is a composite of access lists, global lifetimes, transform sets, and peers that are allowed to create a secure VPN connection. As you can see in Figure 12.11, the crypto maps take information for other areas and join them together. You’ll use the actual command to create this in the step that configures IPSec.

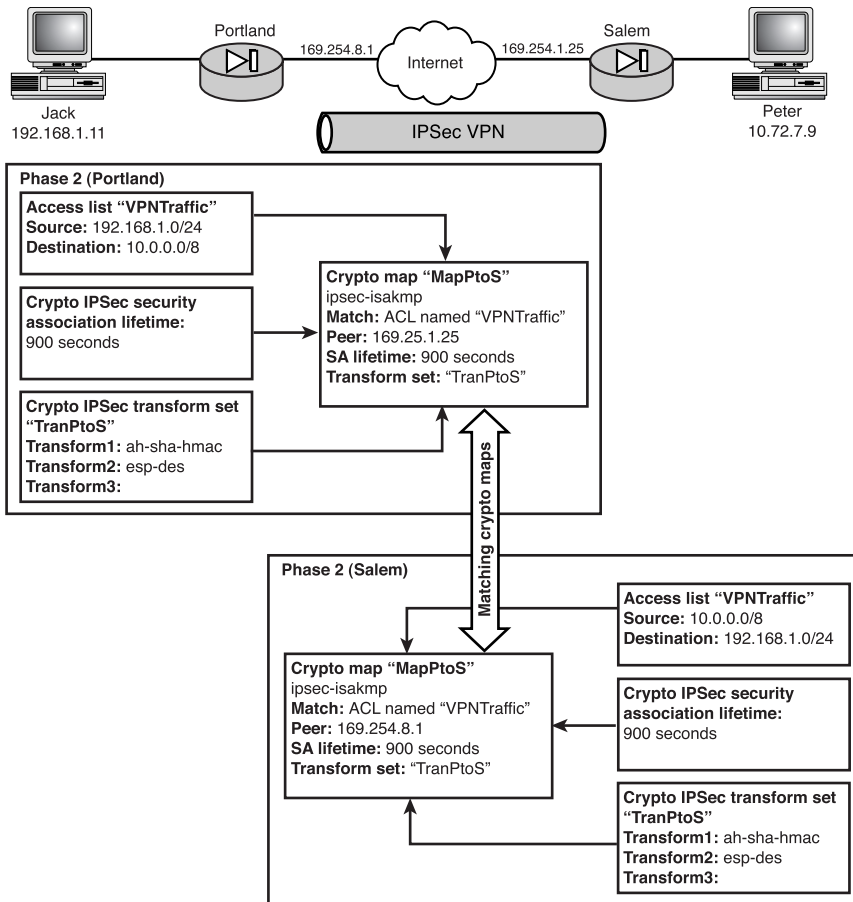
Both of the diagrams in Figures 12.10 and 12.11 are referenced again in the following sections. They are basically your map to constructing a VPN between Portland and Salem.

## Configuring IKE

Now that you have prepared your firewall in the previous step, let’s now begin the configuration stage. The basic steps to configuring IKE, which is also known by the CLI as ISAKMP, are as follows:

1. Enable IKE/ISAKMP.
2. Create an ISAKMP policy.
3. Configure the pre-shared key.





**Figure 12.11** Phase 2 settings.

The commands needed to configure these steps are shown in Listings 12.1–12.3 and are described in the following sections. The command sequence you will need to configure your IKE configuration, as per Figure 12.6, is shown in Listing 12.1.

### Listing 12.1 Portland Firewall

```
portland(config)# isakmp enable outside
portland(config)# isakmp policy 10 authentication pre-share
portland(config)# isakmp policy 10 encryption des
portland(config)# isakmp policy 10 hash md5
portland(config)# isakmp policy 10 group 1
portland(config)# isakmp policy 10 lifetime 86400
portland(config)# isakmp identity address
portland(config)# isakmp key dog address 169.254.1.25
netmask 255.255.255.255
```

Listing 12.2 displays the Salem firewall settings for its ISAKMP policy.

**Listing 12.2 Salem Firewall**

```

saalem(config)# isakmp enable outside
saalem(config)# isakmp policy 10 authentication pre-share
saalem(config)# isakmp policy 10 encryption des
saalem(config)# isakmp policy 10 hash md5
saalem(config)# isakmp policy 10 group 1
saalem(config)# isakmp policy 10 lifetime 86400
saalem(config)# isakmp identity address
saalem(config)# isakmp key dog address 169.254.8.1 netmask 255.255.255.255
    
```

**The isakmp enable Command**

The first and most basic step is to enable IKE/ISAKMP. IKE can be enabled for all interfaces or on a per-interface basis. The command syntax is shown here:

```

pixfirewall(config)# isakmp enable <if_name>
    
```

Table 12.1 displays the isakmp enable options.

**Table 12.1 isakmp enable Options**

Option	Function
if_name	This enables you to specify which interfaces have ISAKMP enabled.

**The isakmp policy Command**

The isakmp policy command enables you to define a group of settings under one priority number. You can create several policies on the PIX firewall if needed; you simply give each group of settings a different priority number. When two peers connect, the lowest policy priority numbers are tried first. So, make your most desired policy the smallest number. The syntax of the isakmp policy command is shown in Listing 12.3.

**Listing 12.3 The isakmp policy Command**

```

pixfirewall(config)# isakmp policy <priority> authn <pre-share|rsa-sig>
pixfirewall(config)# isakmp policy <priority> encrypt <des|3des>
pixfirewall(config)# isakmp policy <priority> hash <md5|sha>
pixfirewall(config)# isakmp policy <priority> group <1|2>
pixfirewall(config)# isakmp policy <priority> lifetime <seconds>
    
```

Table 12.2 displays the available options for the isakmp policy command.

Table 12.2 isakmp policy Options	
Option	Function
<b>priority</b>	This uniquely identifies IKE policy settings to a single group/priority number. You can use numbers between 1 and 65,534.
<b>authen</b>	This defines where to use pre-shared keys or RSA signatures.
<b>encrypt</b>	This defines which encryption algorithm to use in the IKE policy (DES or 3DES).
<b>hash</b>	This defines which hashing algorithm should be used in the IKE policy (MD5 or SHA-1).
<b>group</b>	This defines which Diffie-Hellman group to use: group1 or group2.
<b>lifetime</b>	This specifies how many seconds each SA should exist before new keys are generated. Values are read in seconds and can be between 60 and 86,400 seconds (1 day).

## The isakmp identity Command

When IKE does pre-shared authentication, it needs to associate a pre-shared key with an identity. The peers can identify themselves with either an IP address or a hostname. IP addresses work best in most cases; however, if your peer's address changes often, you should use a hostname instead. The command syntax is as follows:

```
pixfirewall(config)# isakmp identity <address|hostname>
```

Table 12.3 displays the `isakmp identity` options available.

Table 12.3 isakmp identity Options	
Option	Function
<b>address</b>	Indicates that the identity being used is an IP address
<b>hostname</b>	Indicates that the identity being used is a hostname

## The isakmp key Command

The `isakmp key` command creates a pre-shared key and links it to a specific identity. You can have different pre-shared keys per peer if you want by just adding them. If the key is “dog” on one side, it has to be “dog” on the other side because authentication will fail otherwise. The command syntax is shown here:

```
isakmp key <key-string> address <ip> [netmask <mask>]
```

Table 12.4 displays the `isakmp` key options.

Table 12.4 isakmp key Options	
Option	Function
<b>key-string</b>	Specifies the pre-shared key used with an identity. It must match on both ends. The <b>key-string</b> can be up to 128 characters.
<b>ip</b>	Specifies the remote peer's IP address.
<b>netmask</b>	Specifies the remote peer's subnet mask.

## Configuring IPsec

Now that the management connection parameters are set (phase 1), you are ready to set phase 2 parameters. Phase 2 requires several settings combined to create a *crypto map*. This map, as shown in Figure 12.7, uses an ACL named “VPNTraffic”, global SA lifetime settings (optional), and a transform set named “TranPtoS”. These settings are bundled together and attached to an interface. When the two peers negotiate phase 2, they integrate through crypto maps looking for a matching set on both sides. If no match is found, IPsec communication fails. If a match is found, an SA is created and all traffic that matched the ACL in the crypto map is forwarded through the tunnel. The following is an overview of the required tasks:

1. Enable IPsec to enter the firewall.
2. Create a crypto access list.
3. Set the global SA lifetime.
4. Create a transform set.
5. Create a crypto map.
6. Attach the crypto map to an interface.

The commands needed to configure Figure 12.11 are shown in Listing 12.4 and Listing 12.5.

### Listing 12.4 Portland Firewall Crypto Map Settings

```
portland(config)# sysopt connection permit-ipsec
portland(config)#
portland(config)# access-list VPNTraffic permit ip 192.168.1.0
                255.255.255.0 10.0.0.0 255.0.0.0
portland(config)# crypto ipsec security-association lifetime seconds
                28800 kilobytes 4608000
portland(config)# crypto ipsec transform-set TranPtoS ah-sha-hmac esp-des
portland(config)# crypto map MapPtoS 20 ipsec-isakmp
```

**Listing 12.4 Portland Firewall Crypto Map Settings (continued)**

```

portland(config)# crypto map MapPtoS 20 match address VPNTraffic
portland(config)# crypto map MapPtoS 20 set peer 169.254.1.25
portland(config)# crypto map MapPtoS 20 set transform-set TranPtoS
portland(config)# crypto map MapPtoS interface outside
portland(config)# access-list NONAT permit IP 192.168.1.0
                255.255.255.0 10.0.0.0 255.0.0.0
portland(config)# nat (inside) 0 access-list NONAT

```

**Listing 12.5 Salem Firewall Crypto Map Settings**

```

salem(config)# sysopt connection permit-ipsec
salem(config)#
salem(config)# access-list VPNTraffic permit ip 10.0.0.0 255.0.0.0
                192.168.1.0 255.255.255.0
salem(config)# crypto ipsec security-association lifetime seconds
                28800 kilobytes 4608000
salem(config)# crypto ipsec transform-set TranPtoS ah-sha-hmac esp-des
salem(config)# crypto map MapPtoS 20 ipsec-isakmp
salem(config)# crypto map MapPtoS 20 match address VPNTraffic
salem(config)# crypto map MapPtoS 20 set peer 169.254.8.1
salem(config)# crypto map MapPtoS 20 set transform-set TranPtoS
salem(config)# crypto map MapPtoS interface outside
salem(config)# access-list NONAT permit IP 10.0.0.0 255.0.0.0
                192.168.1.0 255.255.255.0
salem(config)# nat (inside) 0 access-list NONAT

```

**The sysopt connection permit-ipsec Command**

To allow IPSec traffic into the firewall, you need to either create several ACL entries permitting protocols 50–51 and port 500 (IKE) or use the `sysopt` command. The ACL option enables you to be granular in specifying which interface will allow IPSec traffic in. However, the `sysopt connection permit-ipsec` command is easier to implement and allows IPSec and L2TP protocol connections on all interfaces. The command syntax is

```

pixfirewall(config)# sysopt connection permit-ipsec

```

**The crypto access-list Command**

Crypto ACL defines which IP traffic should or shouldn't be forwarded through the tunnel. The `crypto access-list` command looks exactly like any other access list but performs in a slightly different way. If traffic matches the `permit` statement in the ACL, it's forwarded and protected by the tunnel. Conversely, if the traffic doesn't match or is denied, the traffic is not dropped but is allowed to travel outside the tunnel. Here is an example:

```

portland(config)# access-list VPNTraffic permit ip 192.168.1.0
                255.255.255.0 10.0.0.0 255.0.0.0

```

This access list states that any source traffic from 192.168.1.0/24 going to destination 10.0.0.0/8 is permitted. Therefore, it is protected by the tunnel.

Note that the only difference between this and other ACLs is that it's used in the `crypto map` command's `match` parameter, making it a `crypto ACL`.

### The `crypto ipsec security-association lifetime` Command

This command is used to set a global SA lifetime value for all `crypto maps` created. The lifetime value is used to define for how long hash and encryption keys are valid. When the lifetime is up, IPsec and IKE generate new keys. Although this is a global command, it can be overridden with a similar command within the `crypto map` commands. The following is the command syntax:

```
pixfirewall(config)# crypto ipsec security-association lifetime seconds
                        28800 kilobytes 4608000
```

Table 12.5 displays the command options for the `crypto ipsec security-association` command.

Table 12.5 security-association lifetime Options	
Option	Function
<b>seconds</b>	This defines the amount of time in seconds for which keys are valid. The default is 28,800 seconds (8 hours).
<b>kilobytes</b>	This defines the amount of data that can pass before the keys are regenerated. The default is 4,608,000KB (10Mbps).



If two peers have different security association values, the lowest value is used. Also, to manually clear all current security associations, the `clear ipsec sa` command can be used.

### The `transform-set` Command

Transform sets define how user data is protected with AH, ESP, or both. A set can contain a maximum of three transforms: one AH for authentication, one ESP for encryption, and one ESP authentication. The `transform-set` command also defines which mode to use—tunnel or transport mode. The command syntax is as follows:

```
pixfirewall(config)# crypto ipsec transform-set <trans-name> [ transform1 ]
                        [ transform2 ] [ transform3 ]
pixfirewall(config)# crypto ipsec transform-set <trans-name> mode transport
```

Table 12.6 displays the command options for the `crypto ipsec transform-set` command.

Table 12.6 transform-set Options	
Option	Function
<b>trans-name</b>	Defines the name of the transform set.
<b>transform1</b>	The first transform (see Table 12.7).
<b>transform2</b>	The second optional transform (see Table 12.7).
<b>transform3</b>	The third optional transform (see Table 12.7).
<b>mode transport</b>	Sets the transform set to transport mode. To make the set tunnel mode, just use the <b>[no]</b> option to turn off transport mode. By default, the mode transport is tunnel mode.

Table 12.7 displays the possible transforms and their uses.

Table 12.7 Transform Options	
Transform	Description
Ah-md5-hmac	Used for authentication
Ah-sha-hmac	Used for stronger authentication
Esp-md5-hmac	Used with ESP-DES or ESP-3DES for additional integrity
Esp-sha-hmac	Used with ESP-DES or ESP-3DES for additional integrity
Esp-des	Used to encrypt with DES (56 bit)
Esp-3des	Used to encrypt with 3DES (168 bit)

Following are two examples of using the `transform-set` command:

```
pixfirewall(config)# crypto ipsec transform-set TranData ah-sha-hmac
```

This example shows a transform set named `TranData` being created with two transforms, using AH with SHA-1 and no encryption.

Here's the second example:

```
pixfirewall(config)# crypto ipsec transform-set
TranData2 ah-sha-hmac esp-des
```

This example shows a transform set named `TranData2` being created with two transforms, using AH with SHA-1 and ESP with DES encryption.



The `crypto ipsec transform-set` command can have only three transforms.

## The crypto map Command

`crypto map` is the command that brings everything together for phase 2. Access lists, lifetimes, transform sets, and peers are all bundled together and given a name and sequence number in a crypto map. Then the map is attached to an interface or multiple interfaces. However, an interface can have only one crypto map assigned to it. Some of the functions the `crypto map` command performs are listed here:

- Defines what traffic is to be protected by IPSec (`crypto acl` and `match` command)
- Designates where the protected traffic should be sent (the remote peer)
- Determines how traffic should be protected (transform set)
- Sets the IPSec security association lifetime (lifetime)
- Specifies the local interface to use for IPSec traffic

`crypto map` commands contain a sequence number that enables you to create multiple entries in a map, which are then iterated from lowest sequence number to highest. Listing 12.6 displays the `crypto map` commands.

### Listing 12.6 crypto map Commands

```
pixfirewall(config)# crypto map <name> <seq_num> ipsec-isakmp|ipsec-manual
pixfirewall(config)# crypto map <name> <seq_num>
    match address <access_list>
pixfirewall(config)# crypto map <name> <seq_num> set peer <IP_address>
pixfirewall(config)# crypto map <name> <seq_num>
    set transform-set <tran_name>
pixfirewall(config)# crypto map <name> <seq_num> set pfs [group1|group2]
pixfirewall(config)# crypto map <name> <seq_num>
    set security-association lifetime
```

Table 12.8 displays the command options for the `crypto map` command.

**Table 12.8 crypto map Command Options**

Option	Function
<b>name</b>	This is the name of the crypto map you are creating.
<b>seq_num</b>	This is the sequence number of the option you are defining. The lowest number is tried first.
<b>ipsec-isakmp   ipsec-manual</b>	These define whether the IPSec SA is defined by IKE ( <b>isakmp</b> ) or manually ( <b>manual</b> ).
<b>match address access_list</b>	This specifies which addresses or crypto ACL address should be protected by the tunnel.



**Table 12.8** crypto map Command Options (*continued*)

Option	Function
<code>peer &lt;ip_address&gt;</code>	This defines the remote peer with which you generating an IPSec tunnel.
<code>tran_name</code>	This defines the transform set to use with the crypt map.
<code>security-association lifetime</code>	This specifies the lifetime for which the keys are valid, and it overrides the global command <code>crypto ipsec security-association lifetime</code> .

## The crypto map interface Command

Now that you have created a crypto map for all your phase 2 IPSec parameters, you need to attach the map to an interface. The `crypto map interface` command can be used to accomplish this, and its syntax is as follows:

```
pixfirewall(config)# crypto map <name> interface <if_name>
```

Table 12.9 displays the options for the `crypto map interface` command.

**Table 12.9** crypto map interface Options

Option	Function
<code>name</code>	This is the name of the crypto map you want to attach.
<code>if_name</code>	Is the name of the interface to which the crypto map is attached.



An interface can have only one crypto map assigned to it, but a crypto map can be used on several interfaces.

## The nat 0 Command

Finally, the last command needed is the `nat 0` command. This command enables traffic from the Portland internal site to travel to the Salem internal site without being NAT translated. The command shown here repeats what is shown in the original commands for the Portland configuration:

```
portland(config)# Access-list NONAT permit IP 192.168.1.0 255.255.255.0
10.0.0.0 255.0.0.0
portland(config)# Nat (inside) 0 access-list NONAT
```

# Testing and Troubleshooting IPsec

After IPsec has been up, the last task is to verify and monitor the connection and parameters. Table 12.10 displays a list of `show` commands used to verify configuration settings. Table 12.11 displays a list of `show`, `clear`, and `debug` commands used to monitor or clear IPsec settings.

**Table 12.10 show Configuration Commands**

Command	Description
<code>show isakmp</code>	This displays the ISAKMP policy settings, similar to the <b>show running config</b> or <b>write terminal</b> command.
<code>show isakmp policy</code>	This displays the default and any other policies created.
<code>show crypt map</code>	This displays the crypto maps created.
<code>show crypto ipsec transform-set</code>	This displays the configured transform sets.
<code>show crypto ipsec security-association lifetime</code>	This displays the global IPsec SA lifetime values.

Table 12.11 displays a list of `show`, `clear`, and `debug` commands used to monitor or clear IPsec settings.

**Table 12.11 show, clear, and debug IPsec Commands**

Command	Description
<code>show isakmp sa</code>	This displays a list of current statuses of IKE security associations.
<code>show crypto ipsec sa</code>	This displays very detailed information about crypto maps assigned to interfaces and traffic flowing across the maps.
<code>clear crypto isakmp</code>	This clears or resets the IKE security associations.
<code>clear crypto ipsec sa</code>	This clears or resets the IPsec security associations.
<code>debug crypto isakmp</code>	This command enables the debug feature of IKE communication between peers.
<code>debug crypto ipsec</code>	This command enables the debug feature between IPsec peers.

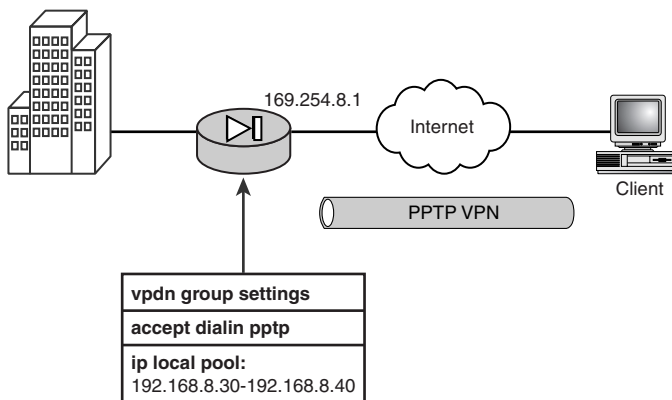


When a PIX firewall is configured to support a VPN tunnel, the VPN tunnel is not created until traffic needs to flow through it, which is similar to interesting traffic on a dial-on demand interface, in which the system does not dial until traffic actually needs to traverse across the line. In addition, several of the PIX firewalls contain VPN lights on their fronts, which show the status of a positive tunnel that has been created.

# Configuring Remote Access Client VPNs

Remote access enables clients to access internal networks via VPN connections to the PIX firewall. You can use the Cisco VPN client software or even use Microsoft's VPN PPTP client to connect the PIX firewall.

When clients connect to the firewall, they must authenticate, and this can be done either locally or via a AAA server. After a client is authenticated, the firewall hands out an IP address, similar to a DHCP server issuing an IP address to the client. This address range is assigned using the `ip local pool` command. Additionally, the PPTP VPN tunnel parameters are set up using the `vpdn` command. This section briefly covers how to allow PPTP clients VPN access into the PIX firewall. Figure 12.12 displays a PPTP VPN client example.



**Figure 12.12** PPTP client example.

The following is a simple list of tasks you must perform:

- Allow PPTP traffic to enter the PIX.
- Create an IP address pool.
- Configure the VPDN group.
- Configure the usernames and passwords.

The commands in Listing 12.7 enable users to access the PIX firewall.

**Listing 12.7 Remote Access Using PPTP**

```

pixfirewall(config)# sysopt connection permit-pptp
pixfirewall(config)# ip local pool pptp-pool 192.168.8.30-192.168.8.40
pixfirewall(config)# vpdn group 1 accept dialin pptp
pixfirewall(config)# vpdn group 1 ppp authentication mschap
pixfirewall(config)# vpdn group 1 ppp encryption mppe 40
pixfirewall(config)# vpdn group 1 client configuration
                        address local pptp-pool
pixfirewall(config)# vpdn group 1 client configuration dns 194.72.6.57
pixfirewall(config)# vpdn group 1 pptp echo 60
pixfirewall(config)# vpdn group 1 client authentication local
pixfirewall(config)# vpdn enable outside
pixfirewall(config)# vpdn username dneuman password 1234

```

## The `sysopt connection permit-pptp` Command

One method of allowing PPTP traffic into the PIX firewall is to use a command similar to the one you used for IPSec. The `sysopt connection permit-pptp` allows PPTP traffic to bypass conduits and ACLs, thus allowing VPN connections into the PIX firewall.

## The `ip local pool` Command

The `ip local pool` command enables you to create a pool of local addresses that are dynamically assigned to remote VPN clients. The following example creates a pool named `pptp-pool` that will allocate addresses from 192.168.8.30 to 192.168.8.40 for remote VPN clients:

```
pixfirewall(config)# ip local pool pptp-pool 192.168.8.30-192.168.8.40
```



The **ip local pool** command is used for remote access VPN clients.

## The `vpdn group` Command

The `vpdn group` command is used to configure and enable L2TP and PPTP remote access VPNs. Table 12.11 describes several options of the `vpdn group` command for a PPTP connection. Listing 12.8 displays the `vpdn group` commands needed to set and configure a PIX firewall for PPTP remote access.

**Listing 12.8 vpdn group Commands Needed for PPTP Remote Access**

```

pixfirewall(config)# vpdn group 1 accept dialin pptp
pixfirewall(config)# vpdn group 1 ppp authentication mschap
pixfirewall(config)# vpdn group 1 ppp encryption mppe 40
pixfirewall(config)# vpdn group 1 client configuration
                        address local pptp-pool
pixfirewall(config)# vpdn group 1 client configuration dns 194.72.6.57
pixfirewall(config)# vpdn group 1 pptp echo 60
pixfirewall(config)# vpdn group 1 client authentication local
pixfirewall(config)# vpdn enable outside

```

**Table 12.11 vpdn group Options**

Command	Description
<b>accept dialin</b>	This defines either L2TP or PPTP as a VPN protocol.
<b>ppp authentication</b>	This specifies the Point-to-Point authentications. The PIX supports PAP, CHAP, and MSCHAP.
<b>ppp encryption</b>	This defines the allowable Microsoft Point-to-Point Encryption. The two types are 40 bit and 128 bit.
<b>client configuration address local</b>	Specifies the local IP address pool for the VPDN group to use for remote clients.
<b>client configuration dns</b>	This specifies the DNS server IP address to hand out to VPN clients.
<b>pptp echo</b>	This specifies the keep-alive timeout value used to keep the VPN tunnel alive.
<b>client authentication local</b>	This specifies where the PIX firewall authentication will occur. AAA services and local databases can be used.
<b>enable outside</b>	This enables PPTP on a single interface.

## The vpdn username password Command

The `vpdn username password` command enables you to create a local list of usernames and passwords for VPDN clients. The command shown here creates a user who can log in using a VPDN group:

```
pixfirewall(config)# vpdn username dneuman password 1234
```

## Scaling VPN Tunnels

Using pre-shared keys works fine in small VPN environments such as the site-to-site configuration. Larger environments connecting several or even hundreds of VPN tunnels use certificate authorities to provide a more scalable solution than pre-shared keys.

When using CAs, each PIX generates its own public and private key pair. The private key stays privately secured on the PIX, whereas the public key eventually is used to create a digital certificate that is utilized during IKE phase 1 to perform authentication. The certificates are validated against the CA before authentication can succeed. This alleviates the need to manually reconfigure all the systems when the keys change, as in the case when using pre-shared keys. The four basic steps needed to configure CAs are as follows:

1. Generate RSA key pairs on the PIX.
2. Obtain the CA's certificate.
3. Request a signed certificate from the CA with the RSA public key generated on the PIX.
4. The CA verifies the request and sends the signed certificate back to be installed on the PIX.

Table 12.12 contains a general list of commands used in this process.

<b>Table 12.12 ca Commands</b>	
<b>Command</b>	<b>Description</b>
<b>ca generate rsa</b>	This generates RSA key pairs, public keys, and private keys. The public key is sent to the CA.
<b>show ca mypubkey rsa</b>	This displays the public key generated.
<b>ca identity &lt;ca_name&gt;</b>	This command creates a name that identifies the IP address for the CA.
<b>ca configure &lt;ca_name&gt;</b>	This configures the CA identity parameters.
<b>ca authenticate &lt;ca_name&gt;</b>	This obtains the CA's certificate.
<b>show ca certificate</b>	This is used to view the CA's certificate.
<b>ca enroll &lt;ca_name&gt;</b>	This enrolls or sends a request to a CA for a certificate.
<b>ca save all</b>	This saves the certificates.