PREV < Day Day Up > NEXT

Chapter 13. IPsec

This chapter covers the following topics:

- Types of IPsec VPNs— This section lays out the two main types of IPsec VPNs.
- Composition of IPsec— This section introduces the protocols that are combined to create the IPsec protocol suite.
- Introduction to IKE— This section discusses some of the basic information about IKE and how it works in the context of the IPsec protocol.
- IPsec Negotiation Using the IKE Protocol— This section talks about how IKE is used to negotiate IPsec tunnels between two peers. Packet-by-packet details of all transactions are given for main mode, aggressive mode, and quick mode negotiations.
- IKE Authentication Methods— This section discusses the IKE authentication methods while focusing on the most important means of authentication available in IKE– Digital Certificates.
- Encryption and Integrity Checking Mechanisms in IPsec— This section covers the details of the various hashing and encryption mechanisms used in IPsec.
- Packet Encapsulation in IPsec— This section talks about how packets are encapsulated in IPsec using ESP or AH in tunnel or transport mode.
- IKE Enhancements for Remote-Access Client IPsec— This section talks about the mode config, x-auth, and NAT transparency features implemented to facilitate remote-access client IPsec.
- IPsec Dead Peer Discovery Mechanism— This section talks about the keepalive mechanism implemented in IPsec.
- Case Studies— This section discusses case studies involving real-life IPsec implementations. Configurations and debugs are included to aid in your understanding of each case study.

IPsec (IP Security) is a suite of protocols used to create virtual private networks (VPNs). According to the IETF, it is defined as follows:

> A security protocol in the network layer will be developed to provide cryptographic security services that will flexibly support combinations of authentication, integrity, access control, and confidentiality.

In the most commonly used scenario, IPsec allows an encrypted tunnel to be created between two private networks. It also allows for authenticating the two ends of the tunnel. However, the IPsec protocol allows only for the encapsulation and encryption of IP data (unlike GRE, which can tunnel but not encrypt non-IP traffic), so to create a tunnel for non-IP-based traffic, IPsec must be used in conjunction with a protocol such as GRE, which allows for the tunneling of non-IP protocols.

IPsec is becoming the de facto standard for creating VPNs in the industry. It has been implemented by a large number of vendors, and interoperability between multivendor devices makes IPsec a very good option for building VPNs. Due to IPsec's growing acceptance and importance in the VPN world, we will discuss it in detail in this chapter.

We will start this chapter by discussing the types of setups IPsec is used to implement. We will then take that basic knowledge and go into a very detailed discussion of the three main protocols in IPsec—IKE, ESP, and AH. We will do a packet-by-packet analysis of how IKE and its component protocols are used to negotiate IPsec tunnels. We will discuss IKE in its various forms, as well as the different authentication methods. This will lead us into a more detailed and separate discussion of the authentication mechanisms available to IPsec. We will look at the advantages and disadvantages of preshared keys, digital signatures, and encrypted nonces. We will then go into a detailed discussion of how encryption and integrity checking (two of the cornerstones of IPsec) are built into IPsec. We will conclude our discussion of IPsec by talking about special enhancements made to IPsec to take care

of remote-access VPN client environments.

PREV < Day Day Up > NEXT
PREV < Day Day Up > NEXT

Types of IPsec VPNs

There are a number of ways to categorize IPsec VPNs. However, it is instructive to look at IPsec in light of the two main VPN design issues it tries to resolve:

- The seamless connection of two private networks to form one combined virtual private network
- The extension of a private network to allow remote-access users (also known as road warriors) to become part of the trusted network

Based on these two designs, IPsec VPNs can be divided into two main categories:

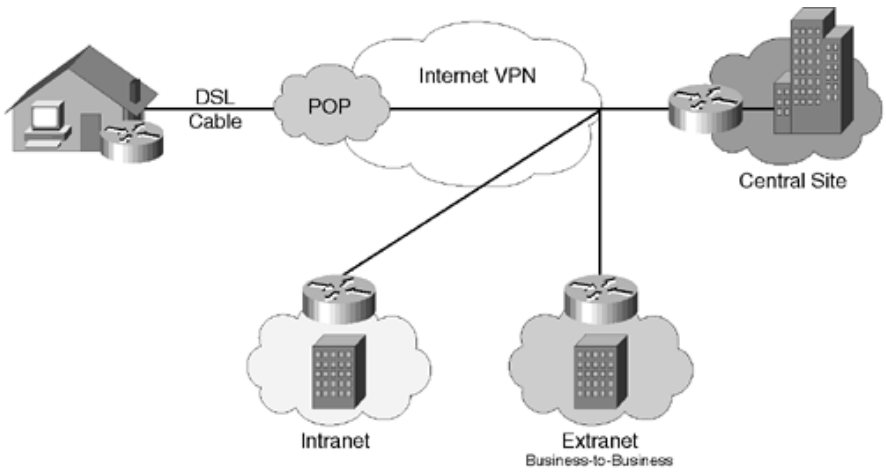
- LAN-to-LAN IPsec implementations (also known as site-to-site VPNs)
- Remote-access client IPsec implementations

LAN-to-LAN IPsec Implementations

LAN-to-LAN IPsec is a term often used to describe an IPsec tunnel created between two LANs. These are also called site to site IPsec VPNs. LAN-to-LAN VPNs are created when two private networks are merged across a public network such that the users on either of these networks can access resources on the other network as if they were on their own private network.

IPsec provides a means of negotiating and establishing an encrypted tunnel between two sites. Generally, tunneling is important, because the two private networks often use RFC 1918 addressing, which needs to be tunneled to traverse the public network. IPsec lets you define what traffic is to be encrypted and how it is to be encrypted. We will look at the details of how IPsec achieves this in the next few sections. Some examples of LAN-to-LAN and site-to-site IPsec setups are shown in Figure 13-1.

Figure 13-1. Types of IPsec VPNs: LAN-to-LAN and Site-to-Site IPsec

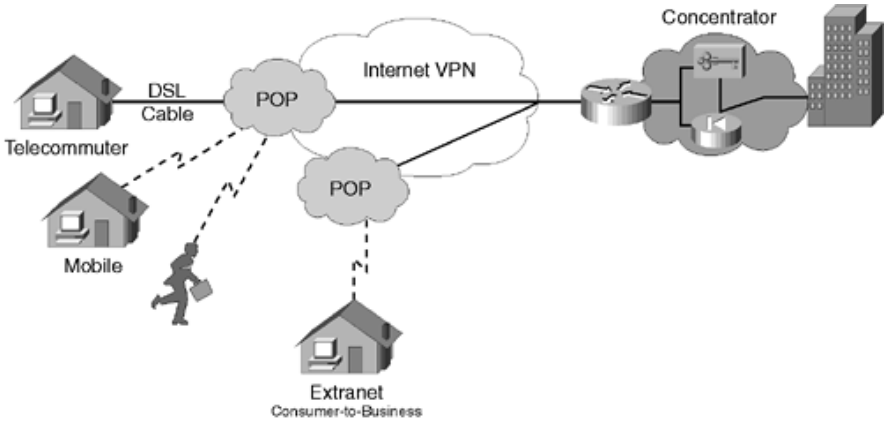


Remote-Access Client IPsec Implementations

Remote-access client IPsec VPNs are created when a remote user connects to an IPsec router or access server using an IPsec client installed on the remote user's machine. Generally, these remote-access machines connect to the public network or the Internet using dialup or some other similar means of connectivity. As soon as basic connectivity to the Internet is established, the IPsec client can set up an encrypted tunnel across the pubic network or the Internet to an IPsec termination device located at the edge of the private network to which the client wants to connect and be a part of. These IPsec termination devices are also known as IPsec remote-access concentrators.

Remote-access implementation of IPsec comes with its own unique set of challenges. In the site-to-site scenario, the number of IPsec peers, meaning the devices that terminate IPsec tunnels, may be limited. However, in the case of remote-access IPsec VPNs, the number of peers can be substantial, even running into the tens of thousands for larger implementations. These scenarios require special scalable mechanisms for authentication key management because maintaining all the keys for all the users might become an impossible task. Later in this chapter we will look at some of the important features that have been added to IPsec to handle remote-access clients setups. Some examples of remote-access IPsec setups are shown in Figure 13-2.

Figure 13-2. Types of IPsec VPNs: Remote-Access IPsec



PREV PREV < Day Day Up > < Day Day Up > NEXT NEXT

Composition of IPsec

IPsec combines three main protocols to form a cohesive security framework:

- Internet Key Exchange (IKE) protocol
- Encapsulating Security Payload (ESP) protocol
- Authentication Header (AH) protocol

Of these three protocols, IKE and ESP are the ones that are mostly deployed together. Although AH is also an important component of the IPsec protocol suite, not that many deployments of IPsec have this protocol turned on for use. In general, much of AH's functionality is embedded in ESP. Therefore, in our discussions in the rest of this chapter, we will focus our attention on ESP, and much of the discussion will assume that we are talking about ESP unless otherwise stated. For example, while discussing quick-mode exchanges in the following sections, we will assume that the goal is to do ESP.

IKE is used to negotiate the parameters between two IPsec peers for setting up a tunnel between them. We will look in detail at the workings of IKE in the next section. ESP provides the encapsulation mechanism for IPsec traffic. We will go into a more detailed discussion of ESP later in this chapter as well.

Table 13-1 describes the three main components of the IPsec protocol suite.

Table 13-1. IPsec Combines Three Main Protocols to Form a Cohesive Security Framework

Protocol	Description
IKE	Provides a framework for negotiating security parameters and establishing authenticated keys.
ESP	Provides a framework for encrypting, authenticating, and securing data.
AH	Provides a framework for authenticating and securing data.

PREV PREV < Day Day Up > < Day Day Up > NEXT NEXT

Introduction to IKE

IKE or Internet key exchange is the protocol responsible for negotiating the IPsec tunnel characteristics between two IPsec peers. IKE's responsibilities in the IPsec protocol include

- Negotiating protocol parameters
- Exchanging public keys
- Authenticating both sides
- Managing keys after the exchange

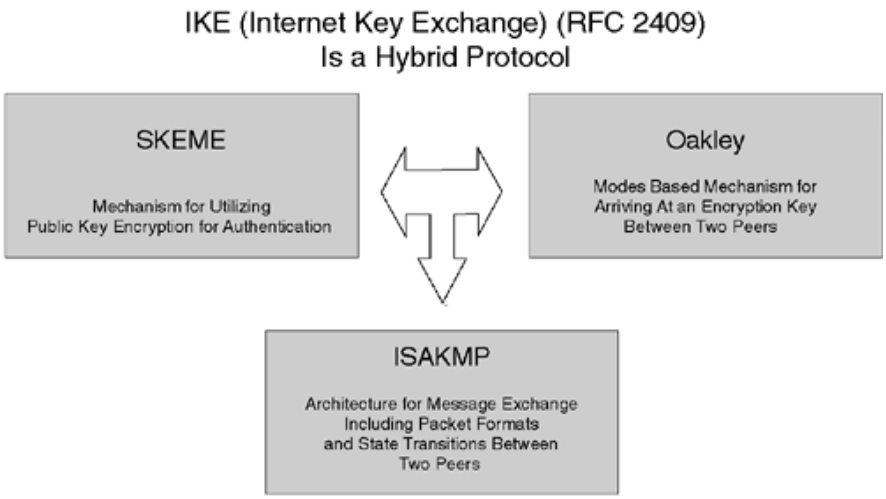
IKE solves the problems of manual and unscalable IPsec implementation by automating the entire key-exchange process. This is one of IPsec's critical requirements.

IKE, like IPsec, is also a combination of three different protocols:

- SKEME— Provides a mechanism for using public key encryption for authentication purposes.
- Oakley— Provides a mode-based mechanism for arriving at an encryption key between two IPsec peers.
- ISAKMP— Defines the architecture for message exchange, including packet formats and state transitions between two IPsec peers.

IKE combines these three protocols in one framework to provide IPsec the facilities just discussed. Figure 13-3 shows these three main components of IKE.

Figure 13-3. Composition of the IKE Protocol



IKE is defined as a standard in RFC 2409. Although IKE does provide a great deal of functionality to the IPsec protocol, some shortcomings in the protocol structure make it difficult to implement in code and scale to new challenges. Work is under way to improve the workings of the IKE protocol and its restandardization in an improved format. This is called the "son-of-IKE" or IKE v2 initiative. See the IETF website for more information on this initiative.

IKE is a two phase protocol. An IPsec tunnel is set up between two peers through the following sequence of events:

Step 1. Interesting traffic is received or generated by one of the IPsec peers on an interface that has been

configured to initiate an IPsec session for this traffic.

Step 2. Main mode or aggressive mode negotiation using IKE results in the creation of an IKE Security Association (SA) between the two IPsec peers.

Step 3. Quick mode negotiation using IKE results in the creation of two IPsec SAs between the two IPsec peers.

Step 4. Data starts passing over an encrypted tunnel using the ESP or AH encapsulation techniques (or both).

Based on Steps 2 and 3, you can see that IKE is a two-phase protocol. Phase 1 is accomplished using main mode or aggressive mode exchanges between the peers, and Phase 2 is accomplished using quick mode exchanges.

This list shows how IKE behaves in a two-phase mechanism:

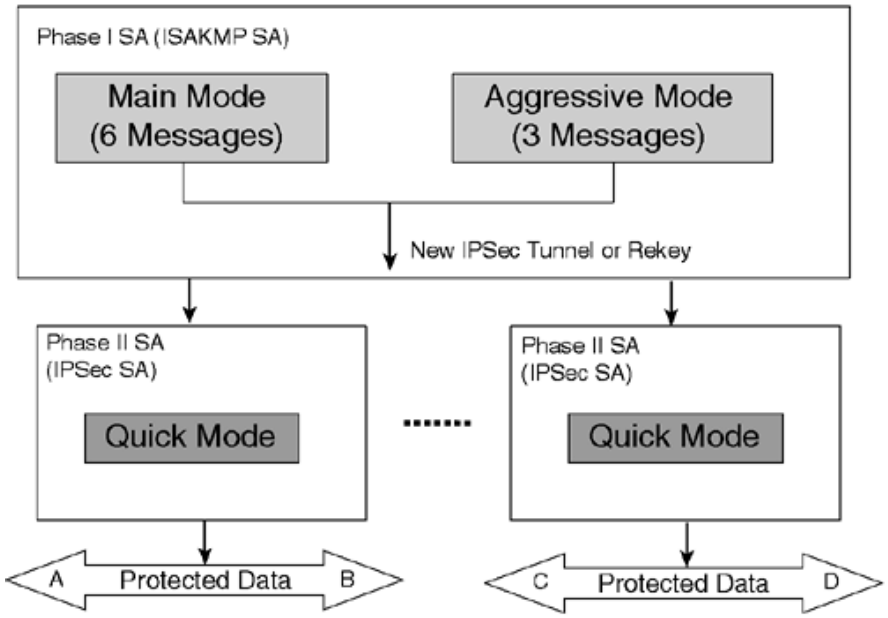
Step 1. In a Phase 1 exchange, peers negotiate a secure, authenticated channel with which to communicate. Main mode or aggressive mode accomplishes a Phase I exchange.

Step 2. In a Phase 2 exchange, security associations are negotiated on behalf of IPsec services. Quick mode accomplishes a Phase II exchange.

Main-mode exchange takes place through the exchange of a total of six messages between the two IPsec peers. If aggressive mode is used, only three messages complete Phase 1 of the exchange. Quick-mode exchange is done using an additional three messages exchanged between the two IPsec peers.

Figure 13-4 shows main mode, aggressive mode, and quick mode in IKE.

Figure 13-4. IKE Main Mode, Aggressive Mode, and Quick Mode



Goals of Main Mode (or Aggressive Mode)

The primary goals of main mode (or aggressive mode) are

- Agreeing on a set of parameters that are to be used to authenticate the two peers and to encrypt a portion of the main mode and all of the quick mode exchange. None of the aggressive mode is encrypted if it is used as the method for negotiation.
- Authenticate the two peers to each other.

- Generate keys that can be used to generate keying material for actual encryption of data as soon as the negotiations have been completed.

All the information negotiated in main mode or aggressive mode, including the keys that are later used to generate the keys to encrypt the data, is stored as what is known as IKE or ISAKMP security association (SA). Any two IPsec peers have only one ISAKMP security association between them.

Goals of Quick Mode

The primary role of quick mode is allow the two peers to agree on a set of attributes for creating the IPsec security associations that will be used to encrypt (in the case of ESP) the data between the two hosts. Also, in the case of PFS (Perfect Forward Secrecy, which you'll read about later), quick mode is responsible for redoing the Diffie-Hellman (DH) exchange so that new keying material is available before the IPsec data encryption keys are generated.

The next section analyzes how IKE is used to negotiate the IPsec parameters between IPsec peers using main (or aggressive) and quick mode.

PREV < Day Day Up > NEXT
PREV < Day Day Up > NEXT

IPsec Negotiation Using the IKE Protocol

IKE negotiates IPsec tunnels between two IPsec peers. This negotiation can be done using a combination of main-mode and quick-mode exchanges or a combination of aggressive-mode and quick-mode exchanges. This section looks at the various packets and message types that are used in these exchanges to do the negotiation. We will look at three types of negotiations that IKE carries out:

- Main mode using preshared key authentication followed by quick-mode negotiation
- Main mode using digital signature authentication followed by quick-mode negotiation
- Aggressive mode using preshared key authentication followed by quick-mode negotiation

In addition to these types, the following types of negotiations can also take place:

- Main mode using encrypted nonces authentication followed by quick-mode negotiation
- Aggressive mode using digital signature authentication followed by quick-mode negotiation

However, we will not go into the details of the last two types because the first three are by far the most common permutations used. We will focus on the most commonly used types of negotiations to aid your understanding.

Main Mode Using Preshared Key Authentication Followed by Quick Mode Negotiation

As stated earlier, this negotiation takes place with the exchange of a total of six messages between the two IPsec peers in main mode followed by three messages exchanged during quick mode.

In the following sections we will walk through the preparation for, as well as the actual exchange of, each message involved in this negotiation.

IKE Phase 1 (Main Mode): Preparation for Sending Messages 1 and 2

The first two messages in the IKE main mode negotiation are used to negotiate the various values, hash mechanisms, and encryption mechanisms to use for the later half of the IKE negotiations. In addition, information is exchanged that will be used to generate keying material for the two peers.

However, before the first two messages can be exchanged, the negotiation's initiator and responder must calculate what are known as cookies. These cookies are used as unique identifiers of a negotiation exchange.

The two peers generate a pseudo-random number that is used for anticlogging purposes. These cookies are based on a unique identifier for each peer (src and destination IP addresses) and therefore protect against replay attacks.

The ISAKMP RFC states that the method of creating the cookie is implementation-dependent but suggests performing a hash of the IP source and destination address, the UDP source and destination ports, a locally generated random value, time, and date. The cookie becomes a unique identifier for the rest of the messages that are exchanged in IKE negotiation.

The following list shows how each peer generates its cookie:

- Generation of the initiator cookie— An 8-byte pseudo-random number used for anti-clogging

  CKY-I = md5{(src_ip, dest_ip), random number, time, and date}

- Generation of the responder cookie— An 8-byte pseudo-random number used for anti-clogging

  CKY-R = md5{(src_ip, dest_ip), random number, time, and date}

IKE uses payloads and packet formats defined in the ISAKMP protocol to do the actual exchange of information. The packets exchanged consist of the ISAKMP header and a series of payloads that are used to carry the information needed to carry out the negotiation.

IKE Phase 1 (Main Mode): Sending Message 1

Let's look at the first message sent across by the IPsec initiator to the respondent and look at the various fields in this message.

Figure 13-5 shows the first IKE message being sent.

Figure 13-5. Sending IKE Main Mode Message 1

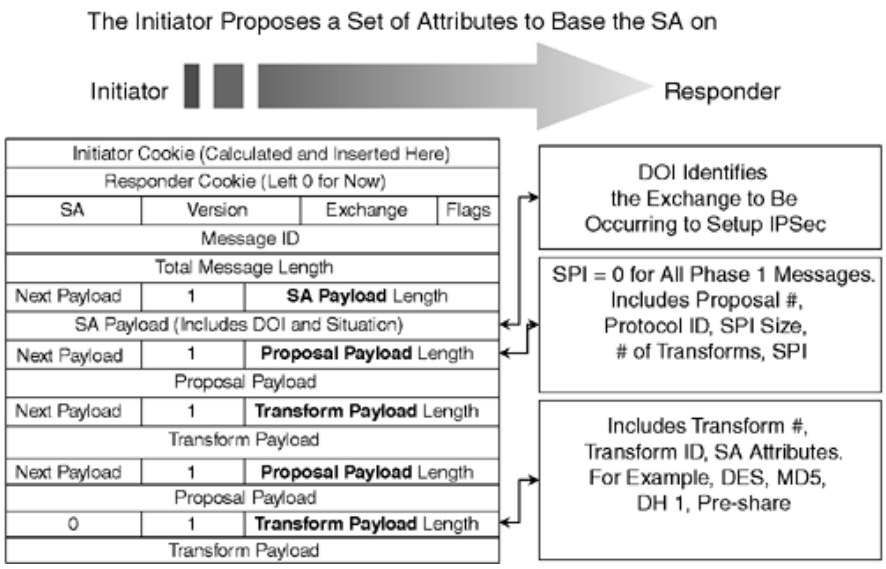


Figure 13-5 shows the ISAKMP header and five payloads. There is one SA payload and two pairs of proposal and transform payloads.

NOTE

All payloads have a field that defines the length of that particular payload. In all the figures in this chapter depicting packet formats, the name of the payload in this field is highlighted to clearly distinguish between the various payloads.

Let's look at each of these in turn:

- ISAKMP header— The ISAKMP header contains the initiator's cookie, and the responder's cookie is left at

0 for the responder to calculate and fill in. The next payload field contains a value signifying that the next payload is the SA payload.

- SA payload— The SA payload contains two important pieces of information. Because ISAKMP is a generic protocol with packet and message formats that can be used to negotiate any number of protocols, it is important to specify that this particular ISAKMP exchange is taking place for IPsec negotiation. Therefore, the SA payload contains a Domain of Interpretation (DOI), which says that this message exchange is for IPsec. The other important piece of information is the situation. The situation definition is a 32-bit bitmask that represents the environment under which the IPsec SA proposal and negotiation are carried out. The situation provides information that the responder can use to make a policy determination about how to process the incoming security association request.

- Proposal payload— The proposal payload contains a proposal number, protocol ID, SPI (security parameter index) size, number of transforms, and SPI. The proposal number is used to differentiate the various proposals being sent in the same ISAKMP packet. The protocol ID is set to ISAKMP, and the SPI is set to 0. The SPI is not used to identify an IKE phase 1 exchange. Rather, the pair of cookies is used to identify the IKE exchange. We will discuss SPI again during phase 2 of IKE where its use is of more significance. The number of transforms indicates the number of transforms that are associated with this particular proposal payload (only one in this case). Note that the packet contains two pairs of proposal and transform payloads.

- Transform payload— Includes transform number, transform ID, and IKE SA attributes. The transform number and the ID are used to uniquely identify the transform from among the rest of the transforms offered in this ISAKMP packet (this sample packet has only one other transform, though). The IKE SA attributes include the attributes that the initiator wants the responder to agree on. These include the type of hash to use in the calculation of various keying materials later in the negotiation, the encryption mechanism to use to encrypt IKE negotiation messages as soon as a shared secret has been established, the Diffie-Hellman exchange mechanism to use, the method of authentication to use, and the timeout values of the IKE SAs negotiated. In this example, assume that the method of authentication being offered is preshared, meaning that a preshared key has been defined on the initiator and responder out of band.

For the sake of this example, assume that the initiator offered the responder the following transform attributes in the first payload. Don't worry about what it offered in the second payload, because we will assume that the responder agrees to what is offered in the first pair of transform and proposal payloads and does not have to worry about the second pair:

- Encryption mechanism— DES
- Hashing mechanism— MD5-HMAC
- Diffie-Hellman group— 1
- Authentication mechanism— Preshared

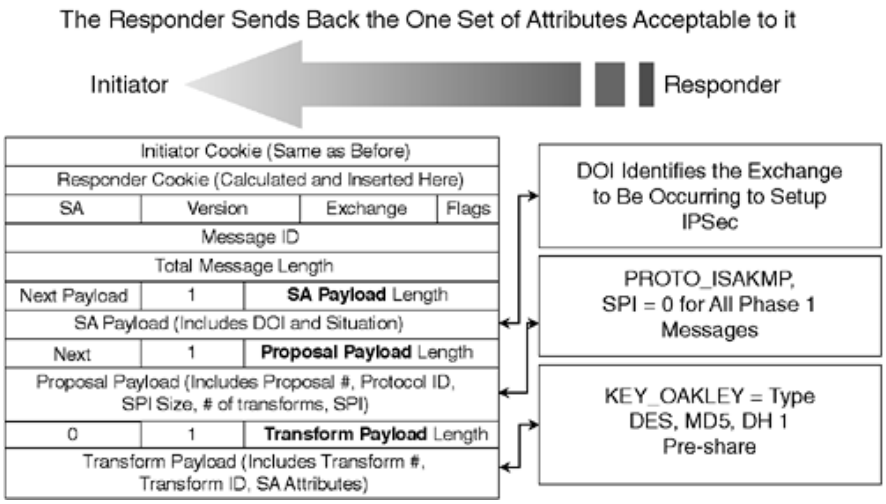
All ISAKMP messages are carried in a UDP packet with destination port 500.

IKE Phase 1 (Main Mode): Sending Message 2

Message 2 is the response from the responder to the packet that was sent by the initiator. Most of the fields are the same as in the packet sent by the initiator, so we will discuss the differences only. Note that only one proposal and transform payload is included in the response because the responder agrees to only one proposal/transform pair and returns that as the agreed-on pair.

Figure 13-6 shows the second IKE message being sent.

Figure 13-6. Sending IKE Main Mode Message 2



Here are the payloads included in this message:

- ISAKMP header— You can see that the ISAKMP header now has both cookie fields set to their respective values.
- SA payload— The SA payload contains pretty much the same material that was sent by the initiator.
- Proposal payload— The proposal payload contains the information for the proposal that the responder has decided to accept.
- Transform payload— The transform payload contains the elements of the transform that the responder has decided to accept.

For the sake of our example, assume that the responder accepted the first proposal described in the preceding section:

- Encryption mechanism— DES
- Hashing mechanism— MD5-HMAC
- Diffie-Hellman group— 1
- Authentication mechanism— Preshared

NOTE

IKE SA timeouts are also negotiated using the transform payloads. The acceptable transform payload must not only have all the other IKE attributes contained in it that are agreeable to the responder, but it also must have an IKE SA timeout value that is equal to or less than the responder that it is set up to accept. If no such transform is included, the negotiation fails.

IKE Phase 1 (Main Mode): Preparation for Sending Messages 3 and 4

The next step that both the initiator and the responder must carry out is to generate material that will be used for the production of a Diffie-Hellman shared secret between the two. Because DH is a critical component of the negotiation taking place between the two peers, we will here discuss how it works.

Diffie-Hellman Algorithm

The Diffie-Hellman algorithm is used in IKE negotiations to allow the two peers to agree on a shared secret, to generate keying material for subsequent use, without knowing any secrets beforehand. (Note that although the preshared secret in this example is already defined on the two peers, the DH secret is used in conjunction with that preshared secret to authenticate the two peers to each other.)

The DH algorithm relies on the following property:

There exists a DH public value = X_a

such that

$X_a = g^a \bmod p$

where

g is the generator

p is a large prime number

a is a private secret known only to the initiator

And there exists another DH public value = X_b

such that

$X_b = g^b \bmod p$

where

g is the generator

p is a large prime number

b is a private secret known only to the responder

Then the initiator and the responder can generate a shared secret known only to the two of them by simply exchanging the values X_a and X_b with each other. This is true because

initiator secret = $(X_b)^a \bmod p = (X_a)^b \bmod p$ = responder secret

This value is the shared secret between the two parties and is also equal to g^{ab}.

Coming back to IKE, in order to calculate the DH secret between the two peers, the two peers calculate the DH public values and send them to each other. In addition, a value known as a nonce is also generated and exchanged. A nonce is a very large random number generated using certain mathematical techniques. It is used in later calculations of the keying material. The following lists describe the preparation for sending message 3 of the IKE.

First, the two peers independently generate a DH public value:

- Generation of the DH public value by the initiator

  DH public value = X_a

  $X_a = g^a \bmod p$

  where

  g is the generator

  p is a large prime number

  a is a private secret known only to the initiator

- Generation of the DH public value by the responder

  DH public value = X_b

  $X_b = g^b \bmod p$

  where

  g is the generator

  p is a large prime number

  b is a private secret known only to the responder

As soon as the DH public values have been calculated, the two peers also independently calculate the nonces:

- Generation of a nonce by the initiator

  initiator nonce = N_i

- Generation of a nonce by the responder

  responder nonce = N_r

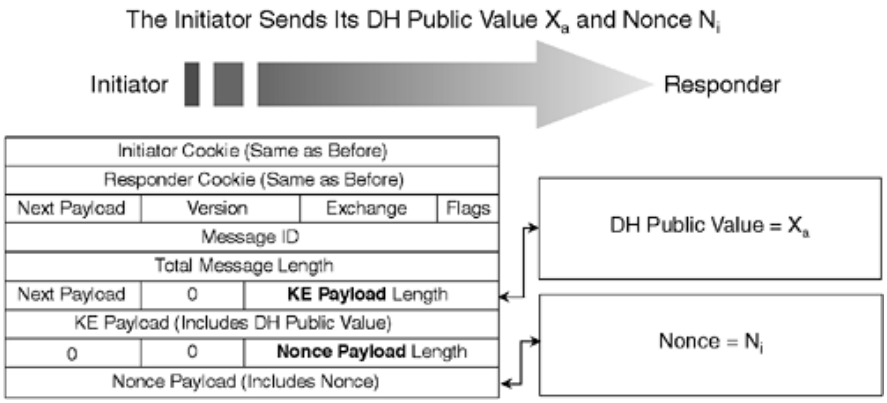
IKE Phase 1 (Main Mode): Sending Message 3

The third message is sent from the initiator to the responder. The ISAKMP header is similar to the one contained in the earlier messages, complete with cookies. However, two new payloads are introduced in this message—the key exchange payload and the nonce payload.

- Key exchange (KE) payload— The KE payload is used to carry the DH public value X_a generated for doing DH.
- Nonce payload— The nonce payload contains the nonce generated in accordance with the preceding discussion.

Figure 13-7 shows message 3 of IKE being sent.

Figure 13-7. Sending Message 3 of IKE

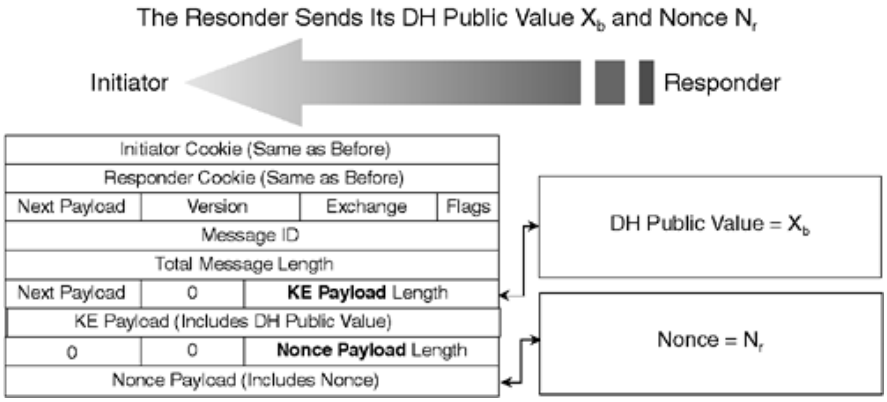


IKE Phase 1 (Main Mode): Sending Message 4

The fourth message is sent from the responder to the initiator. It is very similar to message 3 in that it contains the corresponding DH public value and the nonce for the responder.

Figure 13-8 shows message 4 of IKE being sent.

Figure 13-8. Sending Message 4 of IKE



IKE Phase 1 (Main Mode): Preparation for Sending Messages 5 and 6

Before messages 5 and 6 can be sent, the two peers must calculate the DH shared secret. In addition, based on the exchanged nonce, the DH secret calculated, and the preshared keys stored on the two peers, three sets of keys must be generated that will be used to authenticate the two peers to each other as well as to encrypt subsequent IKE exchanges.

Three keys are generated by each peer. These keys come out to be the same on both ends due to the nature of the DH exchange and the rest of the elements used to generate the keys. These keys are called session keys (SKEYs). The three keys being generated are as follows:

- SKEYID_d— This key is used to calculate subsequent IPsec keying material.
- SKEYID_a— This key is used to provide data integrity and authentication to subsequent IKE messages.
- SKEYID_e— This key is used to encrypt subsequent IKE messages.

Now that both ends have the keying material generated, the rest of the IKE exchange can occur confidentially using encryption done using SKEYID_e.

Note that for the keys to be generated correctly, each peer must find the corresponding preshared key for its peer. A number of preshared keys might be configured on both the peers for each of the many peers they communicate with. However, the ID payload identifying the peer's IP address or the host name does not arrive until the next message is exchanged. Therefore, each peer must find the preshared key for its peer by using the source IP address from which it is receiving the ISAKMP packets. The reason for sending the ID payload later is to hide it by using encryption afforded by the keys that have been generated in this step. We will look at the problems this method can pose in the discussion of aggressive mode.

For the two peers to generate their SKEYIDs, they must first calculate their shared DH secret:

- Calculation of the shared DH secret by the initiator

  shared secret = $(X_b)^a \bmod p$

- Calculation of the shared DH secret by the responder

  shared secret = $(X_a)^b \bmod p$

These two values end up being the same, g^{ab}, as discussed earlier.

After the DH secret has been calculated, the SKEYIDs can be generated. Figure 13-9 shows the session keys being

generated by the initiator, and Figure 13-10 shows the session keys being generated by the responder.

Figure 13-9. Session Keys Being Generated by the Initiator

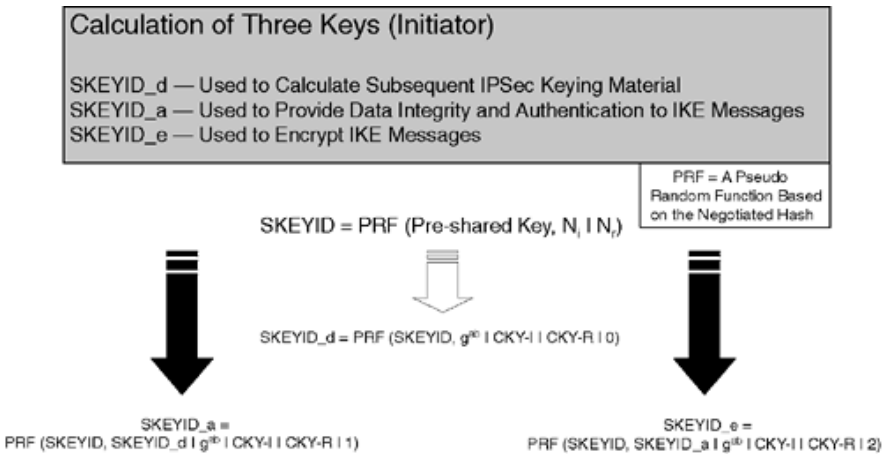
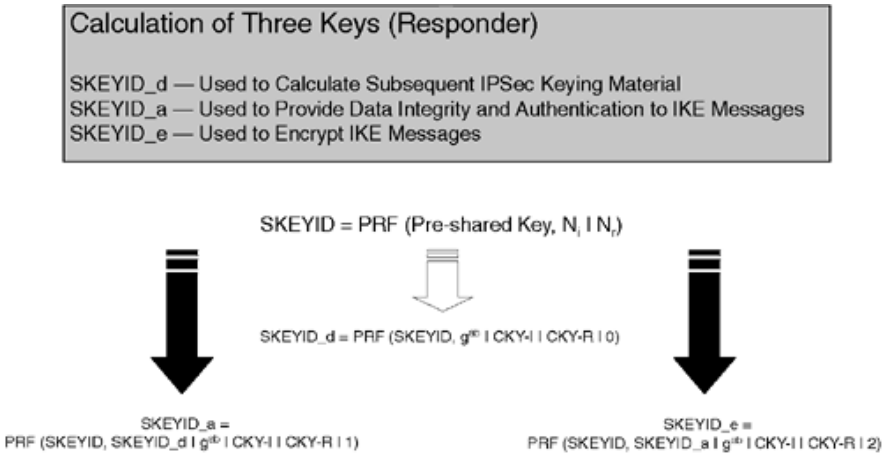


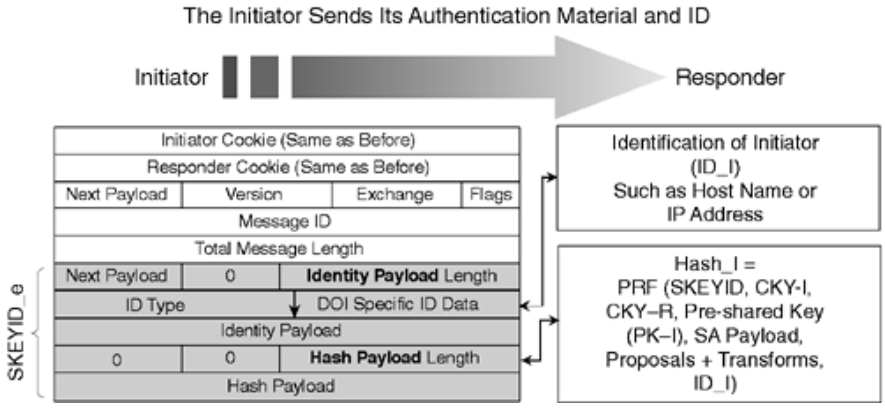
Figure 13-10. Session Keys Being Generated by the Responder



IKE Phase 1 (Main Mode): Sending Message 5

Message 5 is sent with the usual ISAKMP header, but it contains two new payloads—the identity payload and the hash payload. Figure 13-11 shows message 5 of IKE being sent.

Figure 13-11. Sending Message 5 of IKE



Here are the various payloads included in the message:

- Identity payload— This payload contains information about the identity of the initiator. This is in the form of the initiator's IP address or host name.

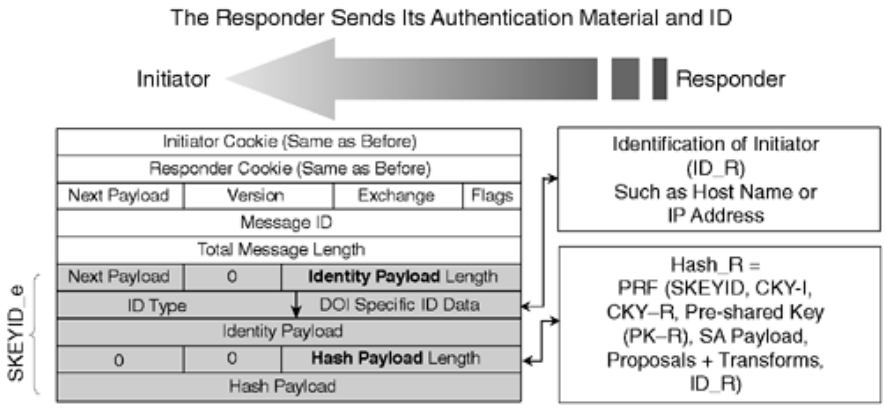
- Hash payload— The hash payload is calculated as shown in Figure 13-11. The hash is used for authentication purposes. The responder calculates the same hash on its end. If the two hashes come out to be the same, authentication is said to have taken place.

The two payloads are encrypted using SKEYID_E.

IKE Phase 1 (Main Mode): Sending Message 6

Message 6 is very similar to message 5 in that the responder sends the corresponding authentication material and its ID. Figure 13-12 shows IKE message 6 being sent.

Figure 13-12. Sending Message 6 of IKE



The contents of the packets are encrypted using SKEYID_E.

Completion of IKE Phase 1 (Main Mode) Using Preshared Keys

The main mode of IKE is completed using the material received in the last two messages of IKE exchange. The two sides authenticate each other by recalculating the hash and comparing it to the one they received in the hash payload. If the two values are the same, the two sides are considered to have successfully authenticated.

The following list shows how the two peers use the authentication hashes received from each other to verify each other's authenticity:

- Initiator authenticates the responder

  Step 1. Decrypt the message using SKEYID_E.

  Step 2. Find the configured PK-R (pre-shared key or responder) using ID_R.

  Step 3. Calculate Hash_R on its own.

  Step 4. If received Hash_R = self-generated Hash_R, authentication is successful!

- Responder authenticates the initiator

  Step 1. Decrypt the message using SKEYID_E.

  Step 2. Find the configured PK-I (pre-shared key or initiator) using ID_I.

  Step 3. Calculate Hash_I on its own.

  Step 4. If received Hash_I = self-generated Hash_I, authentication is successful!

At this point, the IKE SA is established, and main mode using preshared key authentication is complete. The next step, quick mode, is discussed in the next section.

IKE Phase 2 (Quick Mode): Preparation for Sending Messages 1 and 2

Just as the main mode was used to agree upon parameters for the IKE SA, quick mode is used to negotiate some of the parameters of IPsec security association. In our example, we will assume that the initiator has decided to use a property known as Perfect Forward Secrecy (PFS).

PFS

PFS is a property that the initiator of an IKE negotiation can "suggest" to the responder. This suggestion is made by sending a key exchange attribute payload during the first message of quick mode. If the responder agrees to do PFS, it continues the quick mode exchange. Otherwise, it returns "Attributes Not Supported," and the initiator can continue without PFS if it is so configured.

PFS is a property that forces the peers (if they agree to it) to generate a new DH secret during the quick mode exchange. This allows the encryption keys used to encrypt data to be generated using a new DH secret. This ensures added secrecy. If the original DH secret were to somehow get compromised, this new secret can ensure privacy for the data. Also, because the negotiations for the new DH are carried out using encrypted traffic, this new DH secret has an added element of secrecy.

Note that it is not possible to negotiate the DH group the way it was offered and accepted in main mode, because quick mode has only one exchange in which the DH exchange must take place. Therefore, the PFS DH group configured on both ends must match.

To ensure PFS, the two peers generate the numbers needed to perform DH exchange once again. This is done in exactly the same manner as was done in phase 1 of IKE. The following list describes how this is done by the two peers. The apostrophe on top of the various values is used to signify the fact that these values are different from the DH values calculated in phase 1 of IKE:

- Execution of DH by the initiator again to ensure PFS:

  New nonce generated: N_i'

  New DH public value = X_a'

  $X_a' = g^a \bmod p$

  where

  g is the generator

  p is a large prime number

  a is a private secret known only to the initiator

- Execution of DH by the responder again to ensure PFS:

  New Nonce Generated: N_r'

  New DH public value = X_b'

  $X_b' = g^b \bmod p$

  where

  g is the generator

p is a large prime number

b is a private secret known only to the responder

IKE Phase 2 (Quick Mode): Sending Message 1

Message 1 from the initiator to the responder contains payloads you have already seen in the main mode negotiation. The contents of the payloads are, of course, different. Note the presence of the key exchange payload for requesting PFS. In addition, the new nonce is also sent across.

The hash payload contains a hash calculated on some of the elements that were agreed on in phase 1, as well as some of the payloads contained in this message itself. The purpose of the hash is to authenticate the peer again.

In addition, the message contains proposals and transform pairs for the IPsec SAs to be formed. The proposal payload includes the type of encapsulation to use, AH or ESP, and an SPI. The SPI is an interesting element. It is a 32-bit number that is chosen by the initiator to uniquely identify the outgoing IPsec SA that is generated as a result of this negotiation in its database of security associations (it might be talking to a lot more than one peer at a time). The responder, upon seeing the SPI, makes sure that it is not the same as one of the SPIs it is using and starts using it for its incoming IPsec SA. It also proposes an SPI for its outgoing (and the initiator's incoming) SA, which the initiator agrees to after checking.

The associated transform payload includes parameters such as tunnel or transport mode, SHA or MD5 for integrity checking in ESP or AH, and lifetimes for the IPsec security association.

A new payload type introduced in this message is the ID payload. The ID payload contains the proxy identities on whose behalf the initiator does the negotiation. These are generally IP address subnets, but they can have more fields, such as port, too. In the case of a site-to-site IPsec set up with two gateways doing IPsec negotiations with each other, the proxy IDs are based on rules defined on the gateways that define what type of traffic is supposed to be encrypted by the peers.

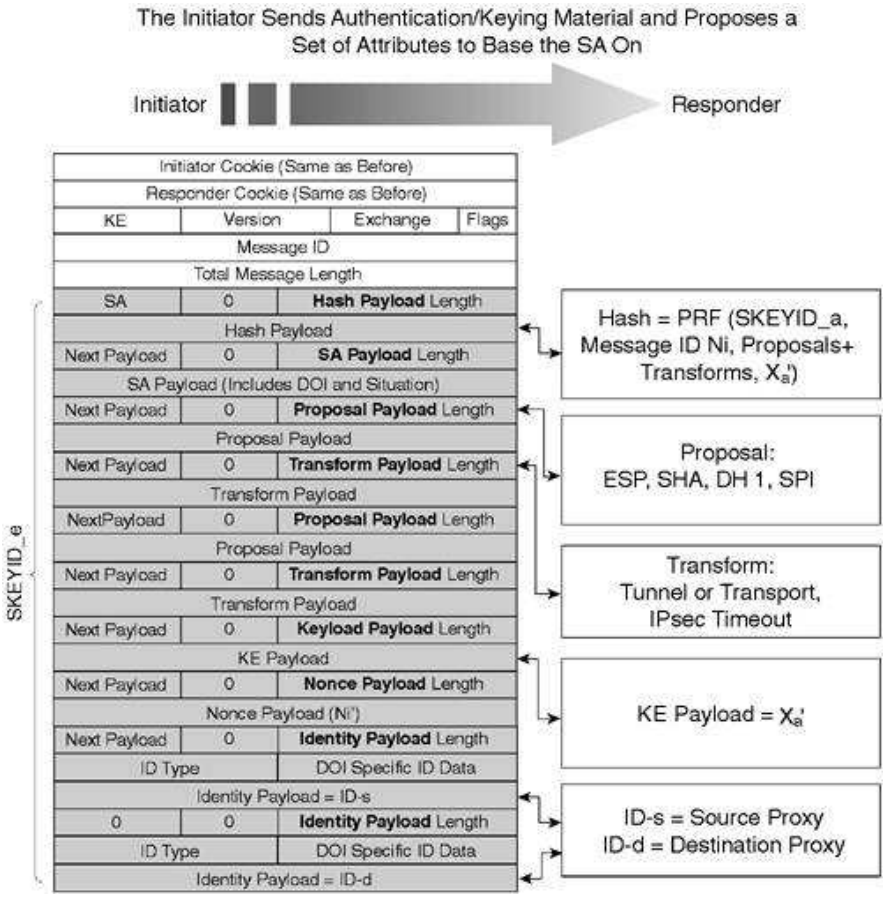
Note that most of the message is still encrypted using one of the keys generated in phase 1.

For the sake of our example, assume that the initiator offers the following transform attributes for the IPsec SA in one of the transform and proposal payload sets:

- Encapsulation— ESP
- Integrity checking— SHA-HMAC
- DH group— 2
- Mode— Tunnel

Figure 13-13 shows the first message of IKE quick mode being sent.

Figure 13-13. Sending the First Message of IKE Quick Mode



NOTE

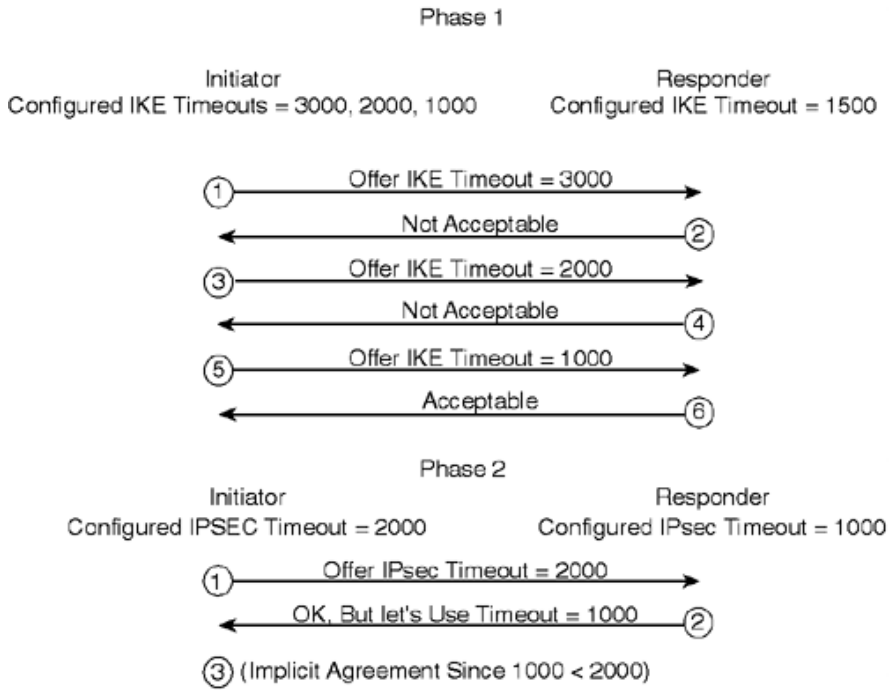
Unlike IKE timeout negotiation, which is very strict, timeout negotiation in quick mode for the IPsec SA is more lenient. If the initiator proposes a transform that, all other things being acceptable, contains a timeout value that is higher than the timeout configured on the responder, the responder does not fail the negotiation. Instead, it simply returns the transform payload with the timeout modified to the lower value configured on it. The initiator then sets up the IPsec with this lower value.

Note that this behavior is different from the behavior shown in the negotiation of the IKE lifetime. A transform with a timeout higher than the one configured on the responder is unacceptable. If none of the transforms carry acceptable attributes, including an equal or lower timeout, the negotiation simply fails.

The difference in the behaviors is due to the fact that the IPsec SA timeout negotiation is done after authentication has taken place, whereas the IKE timeout negotiation is done between unauthenticated peers with no trust established.

Figure 13-14 shows how the lifetimes are negotiated between two peers.

Figure 13-14. IKE and IPsec Lifetime Negotiation



IKE Phase 2 (Quick Mode): Sending Message 2

The second message is sent by the responder in response to the initiator's first message. The fields are pretty much the same, with a few changes. Most notably, only the proposal and the transform payloads that were acceptable are returned to the initiator.

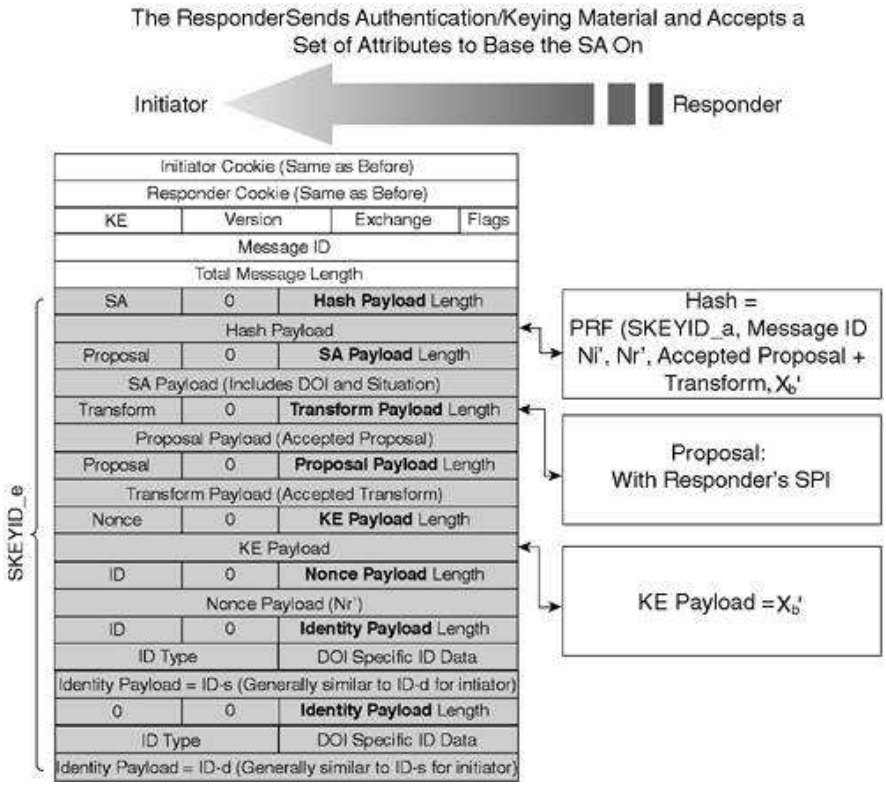
The proposal payload contains the SPI of the outgoing SA for the responder rather than the SPI that was sent by the initiator.

The hash payload contains the hash of only the payloads that are contained in this message. This hash acts as proof of the responder's liveness because it contains a hash of the two latest nonces as well the message ID, proving to the initiator that the responder has indeed received its initial packet and is ready to receive its encrypted messages.

The responder looks at the IDs offered by the initiator in the ID payloads and compares them to the proxy IDs defined on itself. If its policy permits the IDs that are being offered by the initiator, the negotiation continues. Otherwise, a failure occurs.

Figure 13-15 shows message 2 of quick mode being sent.

Figure 13-15. Sending Message 2 of Quick Mode



IKE Phase 2 (Quick Mode): Preparation for Sending Message 3

Before the last message can be sent for quick mode, the two ends must use the information sent relative to DH to generate a new DH secret and use this secret in conjunction with SKEYID_d and some of the other parameters to generate the IPsec encryption/decryption keys. The following are the steps taken by the two peers to generate the keys:

- Initiator generates IPsec keying material

  Step 1. Generate new DH shared secret = $(X_b')^a \bmod p$

  Step 2. IPsec session key for incoming IPsec SA = PRF (SKEYID_d, protocol (ISAKMP), new DH shared secret, SPI_r, N_i', N_r')

  Step 3. IPsec session key for outgoing IPsec SA = PRF (SKEYID_d, protocol (ISAKMP), new DH shared secret, SPI_i, N_i', N_r')

- Responder generates IPsec keying material

  Step 1. Generate new DH shared secret = $(X_a')^b \bmod p$

  Step 2. IPsec session key for incoming IPsec SA = PRF (SKEYID_d, protocol (ISAKMP), new DH shared secret, SPI_i, N_i', N_r')

  Step 3. IPsec session key for outgoing IPsec SA = PRF (SKEYID_d, protocol (ISAKMP), new DH shared secret, SPI_r, N_i', N_r')

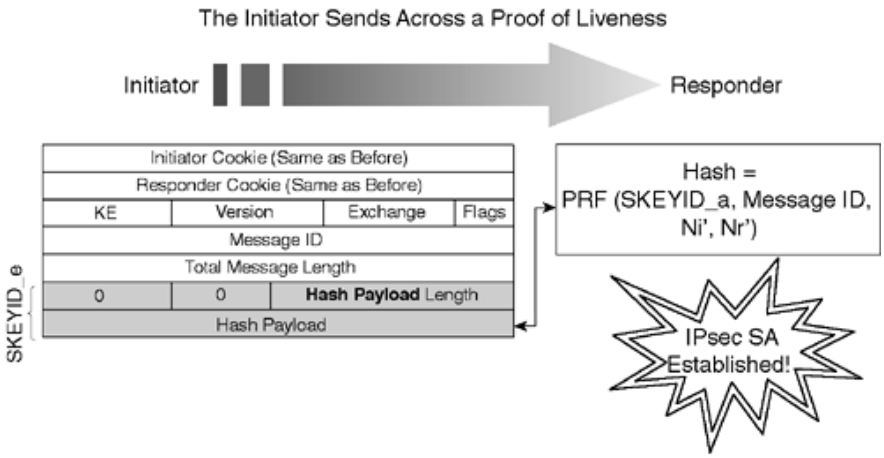
As soon as the IPsec key is generated, the tunnel is pretty much ready to become operational. Only one additional message is sent which concludes the quick mode negotiation.

Please note that two keys are generated on the initiator and the responder: one for the outgoing SA (which matches the key for the incoming SA key on the peer) and one for the incoming SA (which matches the key for the outgoing SA key on the peer).

IKE Phase 2 (Quick Mode): Sending Message 3

The last message of the IKE exchange is sent to verify the liveness of the responder. This is necessary for two reasons. The first reason is that the responder needs some way to know that the initiator received its first and only message of quick mode and correctly processed it. The second reason is to avoid a limited denial of service attack orchestrated by an attacker by replaying a first message of the quick mode exchange from the initiator to the receiver. By sending another message with the correct message ID and latest nonces hashed, the initiator proves to the responder that it has not only received its nonces but also is a live and current peer. Figure 13-16 shows how the final message is sent to the responder.

Figure 13-16. Sending Message 3 of Quick Mode



After message 3 has been received by the receiver, IPsec's quick mode is concluded. All the goals of IKE—authentication, negotiation of the encryption algorithm, and other attributes needed to generate the encrypted packets—have been negotiated. Now the agreed-on IPsec SAs can be used to exchange encrypted traffic.

NOTE

Note that while it is not a requirement per the RFC, the responder can also send the initiator a message similar to the message 3 discussed above to verify its liveliness.

Main Mode Using Digital Signature Authentication Followed by Quick Mode Negotiation

Another important method of authentication used in IKE is digital signatures. We will discuss the use and workings of digital signatures in a later section. Here the changes in main mode of IKE negotiation are described. You might want to read the section on digital signatures (PKI) first and then come back to this section.

The only difference in the preshared and digital signatures method of authentication occurs in the fifth and sixth IKE messages that are sent. The first four messages of main mode and all the quick mode messages remain the same as in preshared key-based IKE negotiation.

Let's look at the preparation and sending of messages 5 and 6 for digital signatures to appreciate the difference.

IKE Phase 1 (Main Mode): Preparation for Sending Messages 5 and 6 Using Digital Signatures

The first difference that is quite evident is the way the SKEYs are generated when using digital signatures. Instead of generating PRF outputs with the preshared secret as one of the components, the PRF output is generated using the DH secret g^{ab} instead, along with the rest of the parameters, which are the same as the ones used in the preshared method. Obviously, at this point, anyone who knows how to do IPsec can negotiate (using a hit-and-miss method to discover configured transforms and proposals) and have the keys generated, which would be valid on both ends. The effect that the preshared secret has, of limiting successful negotiations to those peers that have the same preshared key configured, is not visible in this method up until this point. In order to achieve the same level of control when using digital certificates, a successful negotiation is restricted by either manually setting up each peer with the public key of the peers to which it is allowed to connect or enrolling each peer in a certificate