2

The First Puzzle Piece: The Authentication Header

> It is a riddle wrapped in a mystery inside an enigma.
>
> *Winston Churchill, 1939*

IPsec is an attempt to enable secure communications at the IP layer. This security protection is furnished through the use of two optional headers, the Authentication Header (AH) and the Encapsulating Security Payload header (ESP). Although the use of these headers is optional, their inclusion in IPv6 systems is mandatory; many implementers of IPv4 systems also furnish IPv4 versions of these headers. This chapter describes the AH, its format, its processing, and the protections it provides.

2.1 Protections Provided by AH

AH provides several types of protection [1, 2]:

- *Connectionless integrity* is a guarantee that the message that is received is the exact one that was sent, and that no tampering has occurred. Why "connectionless"? Because communications at the Internet layer are analogous to the Post Office model rather than the phone company model. Messages are sent from the sender to the

receiver, but no attempt is made to ensure that they are received in order or that any (or all) were in fact received. That task is left to the transport layer protocol or to the application that originates the messages.

- *Data origin authentication* is a guarantee that the message actually was sent by the apparent originator of the message and not by another user masquerading as the supposed message originator.
- *Replay protection* (optional) is the assurance that the same message is not delivered multiple times and that messages are not delivered grossly out of order. This capability must be implemented by the sender; the receiver may optionally enable its use.

2.2 Security Associations and the Security Parameters Index

Before two communicating entities can exchange secure communications, they need to agree on the nature of the security to be applied to those communications: which security headers (AH, ESP, or both) will be applied, the cryptographic algorithms to be used, the secret keys, and so forth. A security association (SA) consists of all the information needed to characterize and exchange protected communications. The IETF documents treat the SA and its repository, the security association database (SAD) as hypothetical constructs, because they are entities that are internal to each of the peers. They contain information essential to conducting secured communications via the IPsec protocols, but the SA in its entirety is not part of that communication, so the documents do not dictate its form or location. In practice, the SAD generally is a table that is kept in protected storage by the system process that handles these communications.

Each SA includes various pieces of information that the IPsec-processing routines can use to determine whether the SA is eligible to be applied to a particular inbound or outbound message. Each such item can have a specific value or values, to narrowly define those messages to which the SA applies; or a wildcard value, to indicate that an item is not relevant in evaluating traffic for the SA. These items, called the SA's selectors, include the following:

- *Source and destination addresses* (IPv4 or IPv6). Each of these addresses can be a single IP address: unicast, anycast (IPv6 only), broadcast (IPv4 only), or multicast; a range of addresses; an address plus mask, to specify a subnet. For a single SA, the source address(es)

and the destination address(es) all must be either IPv4 or IPv6. If the sole selectors for an SA are the IP addresses of the communicating peers, the SA is called a *host-oriented* SA, because it governs all communications between the two systems, regardless of which users or applications are involved.

- *Name*, either a user ID or a system name. The User ID limits this SA to traffic initiated by or destined for a specific user. If the sole selectors for an SA are the user IDs of the communicating peers, the SA is called a *user-oriented* SA, because it governs all communications between the two users, regardless of which systems or applications are involved. The system name limits it to traffic for a specific system, which can be a host, a security gateway, or any other addressable system. The system name can be specified in one of the following three formats; the user ID can be specified in one of the first two formats:
  - A fully qualified DNS user name (e.g., frankel@artechhouse.com) or DNS system name (e.g., artechhouse.com);
  - An X.500 distinguished name (explained in Chapter 10);
  - An X.500 general name (explained in Chapter 10).
- *Transport Layer Protocol* (TCP or UDP).
- *Source and destination ports*. A single port number generally is used to limit the SA's applicability to a single type of application traffic (e.g., FTP or TELNET). When one or both of the port selectors are used in combination with the Transport Layer Protocol selector and one or both of the address selectors, the SA is called *session-oriented*, because its effect is to limit the SA to one session, or instantiation, of a particular type of traffic between two specific hosts.

Each SA also contains various pieces of information that must be made available to the IPsec-processing routines, including:

- Data used to provide authentication protection: AH or ESP authentication algorithm, keys, and so forth (further explained later in this chapter and in Chapter 4);
- Data used to provide confidentiality protection: ESP encryption algorithm, IV, keys, and so forth (described in Chapters 3 and 4);
- Data used to provide anti-replay protection: sequence number counter and sequence counter overflow flag for outbound SAs,

anti-replay counter and anti-replay window for inbound SAs (further explained later in this chapter);

- IPsec header mode flag: Tunnel Mode, Transport Mode, or both (further explained later in this chapter);
- SA lifetime, measured in elapsed time or number of bytes protected (SA expiration and replacement are discussed in Chapter 5);
- Data used to perform message fragmentation: PMTU information for outbound SAs (further explained later in this chapter).

The granularity of an SA is a rough measure of the SA's selectivity. An example of an SA with a coarse granularity could be a host-to-host SA or even a network-to-network SA, one that applies to all traffic between the two hosts or networks, regardless of application or user. An SA with a moderate granularity might be limited to a specific type of traffic between two hosts, such as FTP, or to all traffic between two hosts conducted by a specific user on each host. An example of an SA with a fine granularity is one that could be limited to a specific session between two hosts, such as a single FTP file transfer session.

It is highly likely that multiple SAs will be established between a pair of communicating hosts. For example, one set of security features might be required for email or Web communications and a different, more stringent set for a remote payroll application. When protected messages are sent, the sender needs to indicate which SA was used to encode the communication, so the receiver can use the same SA in decoding the message. That is the function of the security parameters index (SPI). Because each SA is unidirectional, protected two-way communications between two peers requires the establishment of two SAs: an inbound SA and an outbound SA. The SPI, in conjunction with the destination address and the security protocol (AH or ESP), is sufficient to unambiguously select a unique inbound SA from the SAD. To ensure the SPI's uniqueness, each peer selects the SPI for its own inbound SA.

Another hypothetical database, the security policy database (SPD), reflects more general policies governing the treatment of various classes of protected and unprotected traffic. Each SPD entry can result in the creation or negotiation of one or more SAs. The SPD is discussed in excruciating detail in Chapter 9; for now we simply assume that there is a magical policy mechanism that is used to determine which SA (if any) applies to an

incoming or outgoing message; we also assume that the applicable SA has been added, *deus ex machina* (more on that in Chapter 5), to the SAD.

2.3 AH Format

Figure 2.1 illustrates the AH format. The header comprises six fields. Five of the fields have a fixed length, for a total length of three 32-bit words; the sixth field is variable length. The individual header fields are as follows:

- *Next header* is the type of the header that follows the AH. It might be the other IPsec header, the ESP header; a TCP header if the application that originated the message runs over TCP (e.g., email or Web access via HTTP); a UDP header if the originating application runs over UDP (e.g., the troubleshooting program traceroute); or an ICMP header, if this is an IP error or informational message. In IPv6, it could be one of the extension headers.
- *Payload length* is the length of the total AH in words, minus 2 (or the length of the authentication data portion of the header, plus 1). This elegant calculation is a legacy of the former version of the AH, defined in RFC 1826, which did not include a mandatory Sequence Number field. The intent is to transmit the length of the authentication data, which is a variable-length field, to the receiver. Initially, an optional sequence number was included in the authentication data, and the Payload Length field conveyed the length of that combined field. Once the Sequence Number field was made mandatory and was separated from the Authentication Data field, a graceful description of the Payload Length field became impossible.
- *RESERVED* is a field currently set to 0 but reserved for future use.
- *Security parameters index* (SPI) is the index into the receiver's SA database.

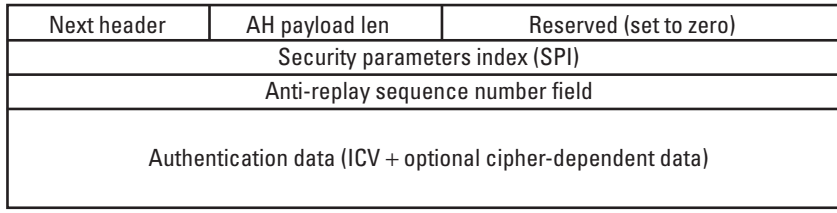


Figure 2.1 AH format.

- *Sequence Number field* is the number of messages sent from the sender to the receiver using the current SA. By keeping track of this quantity and sending it to the receiver, the sender enables the receiver to perform replay protection, if desired.
- *Authentication Data field* is a variable-length field that fulfills the AH's main purpose. It contains the integrity check value (ICV), which is a cryptographic version (more on this in Chapter 4) of the message's contents that can be used by the receiver to check the message's authentication and integrity. This field is padded, if necessary, so that the total length of the AH is an exact number of 32-bit words (IPv4) or 64-bit words (IPv6).

2.4 AH Location

Figure 2.2 illustrates AH's placement for both IPv4 and IPv6. In IPv4, it follows the IP header, preceding the next header (ESP, TCP, UDP, or ICMP). Nothing else intervenes between the AH and its preceding IP header or its trailing next header. In IPv6, the positioning of AH is similar, but the optional IPv6 extension headers can either precede or follow AH. The IPv6 extension headers that can precede AH are the hop-by-hop header, the

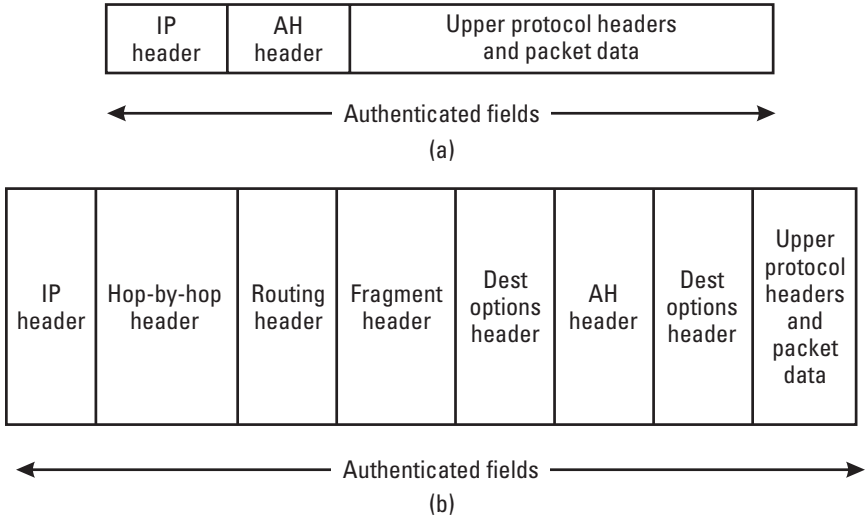


Figure 2.2 AH placement in Transport Mode: (a) IPv4 and (b) IPv6.

routing header, and the fragment header. The destination options header can either precede or follow AH. Its position relative to AH is dependent on whether the special processing should take place before or after authentication processing occurs.

2.5 AH Modes

An additional factor governs the placement and processing of AH. Figure 2.2 illustrates the placement of AH in what is known as Transport Mode. This mode is used primarily for end-to-end authentication between two hosts. However, when a security gateway is used to provide protection for multiple hosts on a network, Tunnel Mode is used. An additional (outer) IP header, whose source address is that of the security gateway, is placed at the beginning of the packet; the original (inner) IP header, whose source address is one of the network hosts protected by the gateway, is left intact. The new IP header's destination address can be the same as the original IP header's destination address, or, if the destination is also protected by a security gateway, the new IP header's destination address can differ from the original IP header's destination address. Figure 2.3 illustrates AH's placement in Tunnel Mode. In IPv4, AH follows the new IP header and precedes the original IP

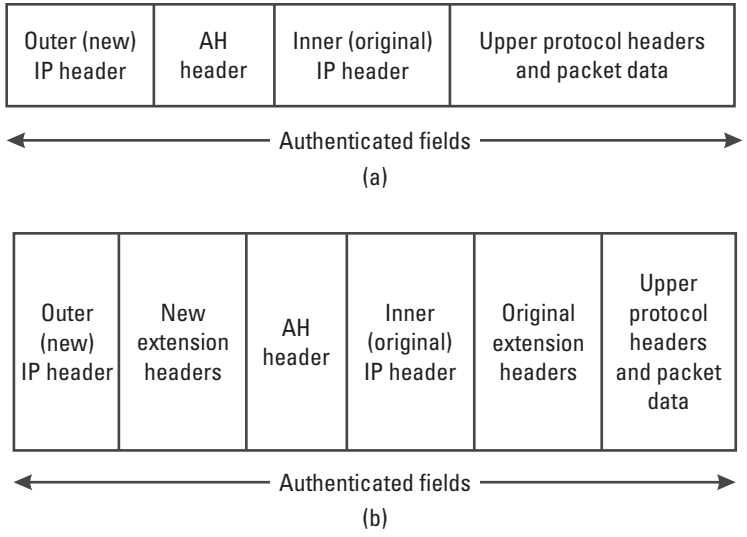


Figure 2.3 AH placement in Tunnel Mode: (a) IPv4 and (b) IPv6.

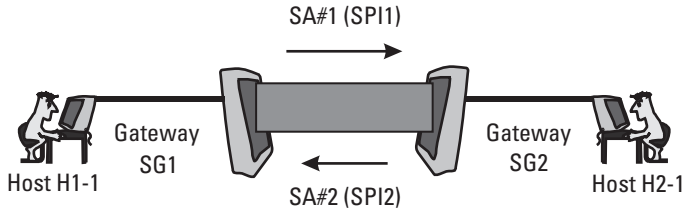
header. In IPv6, AH follows the same extension headers (if present) that it follows in Transport Mode and precedes the original IP header.

Tunnel Mode also can be used for host-to-host communications, in which case the addresses are the same in both the original IP header and the additional IP header.

In scenario 1, to provide AH protection between hosts H1 and H2, either Transport Mode or Tunnel Mode could be used. In scenario 2, if gateways SG1 and SG2 are to provide AH protection for their hosts, a Tunnel Mode SA will be established between SG1 and SG2. Figure 2.4 illustrates the Tunnel Mode communication. Traffic from H1-1 to H2-1 will traverse the leg from SG1 to SG2 inside a Tunnel Mode packet whose outer header has source address SG1 and destination address SG2, but whose inner header has source address H1-1 and destination address H2-1. In scenario 3, if gateway SG2 is to provide AH protection for its hosts, a Tunnel Mode SA will be established between host H1 and SG2. Traffic from H1 to host H2-1 will traverse the leg from H1 to SG2 inside a Tunnel Mode packet whose outer header has source address H1 and destination address SG2, but whose inner header has source address H1 and destination address H2-1. Figure 2.5 illustrates the Tunnel Mode headers for each of the three scenarios.

2.6 Nested Headers

More than one SA can be applied to a single message. If both endpoints of both SAs are the same, the AHs are referred to as adjacent AHs; if one or both sets of endpoints differ, the AHs are referred to as nested AHs. Adjacent



SA #	Src addr	Dest addr	IPsec protocol	SPI	Mode
1	SG1	SG2	AH	SPI1	Tunnel
2	SG2	SG1	AH	SPI2	Tunnel

Figure 2.4 Tunnel Mode SA between gateways.

Outer IP header	AH header	Inner IP header
Source: H1 Destination: H2		Source: H1 Destination: H2

Outer IP header	AH header	Inner IP header
Source: SG1 Destination: SG2		Source: H1-1 Destination: H2-1

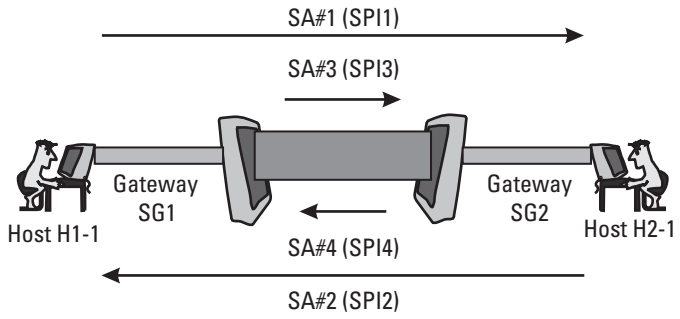
Outer IP header	AH header	Inner IP header
Source: H1 Destination: SG2		Source: H1 Destination: H2-1

Figure 2.5 Sample tunnel headers.

AHs do not provide extra protection, and their implementation is not mandated. (Adjacent IPsec headers are discussed in Chapter 3.) Nested AHs do make sense in certain contexts. In scenario 2, if host H1-1 and host H2-1 require end-to-end authentication, but each is protected by a security gateway that demands to authenticate all traffic transiting the gateway, nested AHs are a reasonable approach to fulfill both requirements. A Tunnel Mode SA can protect traffic between SG1 and SG2, and a Transport Mode SA can protect traffic between H1-1 and H2-1. When a message is sent from H1-1 to H2-1, it will have a single Transport Mode AH from the time it leaves H1-1 until it arrives at SG1; as it travels from SG1 to SG2, it will incorporate nested AHs, an inner Transport Mode AH, and an outer Tunnel Mode AH; and traveling from SG2 to H2-1, it will once again have a single Transport Mode AH. Figure 2.6 illustrates the four unidirectional SAs, each with its own SPI, that provide this protection.

2.7 Implementing IPsec Header Processing

Generally, one portion of the operating system, or kernel, is responsible for networked communications. For outbound messages, the networking routines add the IP header to the message, fragment messages when needed, and



SA #	Src Addr	Dest Addr	IPsec Protocol	SPI	Mode
1	H1-1	H2-1	AH	SPI1	Transport
2	H2-1	H1-1	AH	SPI2	Transport
3	SG1	SG2	AH	SPI3	Tunnel
4	SG2	SG1	AH	SPI4	Tunnel

Figure 2.6 Nested AH SAs.

forward the message to the network access or physical layer to be sent out. For inbound messages, the routines accept messages from the network access layer, reassemble fragmented messages when appropriate, strip off the IP header, and forward the message to the transport or application layer for further processing. How do the IPsec-processing routines fit in relative to the operating system networking logic? There are three common approaches:

- Modifying the networking (IP stack) code. This is the most direct approach, but it involves a change to the kernel code, so it would normally be the solution of choice for developers of operating systems. It is applicable to both hosts and gateways.
- Separating the IPsec code from the networking code. This approach does not involve changing the kernel code, but it can necessitate reimplementing portions of the networking code (e.g., fragmentation and reassembly of messages). It generally is referred to as a "bump-in-the-stack" (BITS) implementation, because the IPsec code is placed between the Internet layer of the stack and the network access layer. A BITS implementation is applicable to hosts and gateways, but it is more commonly found on hosts with legacy operating systems.

- Placing the IPsec code outside the machine. This external crypto-processor, referred to as a "bump-in-the-wire" (BITW) implementation, is the least intrusive option, in terms of the kernel code. The IPsec code can be integrated with router or firewall code and placed in a router or firewall, or it can be implemented in a standalone "IPsec box." It can be attached to a single host or gateway or to multiple machines.

2.8 AH Processing for Outbound Messages

Once it has been determined that an outgoing message needs the protection afforded by AH (more on that in Chapter 9), and the outbound SA governing the protected communication has been found (more on that in Chapter 9) or negotiated (described in Chapter 5), the message is passed to the IPsec-processing routines, which perform the following steps.

1. Insert an AH template in the proper place (as described above).
2. Fill in the Next Header field.
3. Fill in the SPI field with the SPI of the selected SA.
4. Compute the Sequence Number field. This field has a length of 32 bits, which means it can hold a maximum value of 4294967295 (hex FFFFFFFF, or 2^{32-1}). If the selected SA has been used to protect less than that number of messages, the Sequence Number field is simply incremented by 1; the new value is placed in the AH and also saved in the SAD. However, if the Sequence Number field has reached its maximum value, meaning that this SA has already been used to protect the maximum allowable number of messages, there are several possibilities. If the SA's secret keys were negotiated by the peers (more on that in Chapter 5), it is time to negotiate new keys, whether or not the message recipient has enabled replay protection. This message is set aside or discarded until that can take place. If the SA's keys are manually established keys that were agreed on by the peers in some unspecified manner (e.g., over the telephone or through the use of couriers), and if the sender knows that the recipient is not enabling replay protection, the sequence number is simply reset to 1. For manually established keys, in the case where the recipient does require replay protection, new keys

must be agreed on. Until that happens, this message cannot be sent, and the AH processing comes to a halt.

5. For a Transport Mode SA, change the preceding IP header's Next Header field to AH.
6. Add a tunnel header, if required. If the SA specifies Tunnel Mode, the additional (outer) IP header must be constructed and added to the message. The source and destination addresses of the outer header are the tunnel endpoints, as specified by the SA.

   If both headers are IPv4 headers, the following fields are copied from the inner header to the outer header: Version, Type of Service, Protocol, Fragment Identification, MF (More/Last Fragment) Flag, and Fragment Offset. The following fields are recomputed for the outer header: Header Length, Total Length, and Header Checksum; the recomputation is necessary so that these fields incorporate information from both the inner and outer IP headers and from the AH. The Next Header field is set to AH. The Options field is not copied. The TTL is set to the system's default value. The local system's policy also determines the value of the DF (don't/may fragment) Flag: It can be copied from the inner header, set to 1 to prohibit fragmentation, or set to 0 to allow fragmentation. The fields of the inner header are left intact, with the following exception: If the source addresses of the inner and outer headers differ, that means the inner packet has traveled to reach the tunnel's source address. In this case, the inner header's TTL field is decremented and the inner header's Header Checksum is recomputed to reflect that change.

   If both headers are IPv6 headers, the following fields are copied from the inner header to the outer header: Version and Traffic Class. The Payload Length field is recomputed for the outer header; the recomputation is necessary so that this field incorporates the lengths of both the inner and outer IP headers and the AH. The Next Header field is set to AH or to the header type of the extension header that precedes the AH. The extension headers themselves are not copied. The hop limit is set to the system's default value. The fields of the inner header are left intact, with the following exception: If the source addresses of the inner and outer headers differ, that means the inner packet has traveled to reach the tunnel's source address. In that case, the inner header's Hop Limit field is decremented.

If the inner header is an IPv4 header and the outer header is an IPv6 header, or vice versa, the processing is slightly different: The Version field is set to 4 for the IPv4 header and to 6 for the IPv6 header; the Traffic Class field is transformed into TOS; and the source and destination addresses are converted to the appropriate format, if necessary.

7. Compute the authentication data. The authentication data consist of the output of a keyed message hash. An algorithm (more on this in Chapter 4) is used that takes a message of any size and generates a fixed-length output, with the property that it is infeasible to modify a message in such a way that the resulting hash of the modified message would be equivalent to that of the original message. Incorporating a secret key into the hash computations makes it impossible for a user not privy to the key to fake an authenticating hash.

   The entire message is not protected by the AH, because IP headers can contain three classes of data: immutable data, which never changes in transit; mutable but predictable data, which can be modified during transit, but whose final value, on arrival at the destination, is predictable; and mutable unpredictable data, whose value can change during transit in an unforeseen manner. Table 2.1 lists the fields of the IP header that fall into each category. Only the message data and those header fields that will not change in an unpredictable manner in transit are used as input to the authenticating hash, so the final recipient of the packet can verify the hash. Thus, in Transit Mode, the message data and the predictable fields of the IP header are protected. In Tunnel Mode, the entire original IP header and the message data are protected, but only the predictable fields of the added header are protected. When the hash is computed, zeroes are used in place of the contents of the unprotected header fields.

   The mandatory keyed hash algorithms for IPsec AH are HMAC-MD5, which generates a 128-bit hash, and HMAC-SHA-1, which generates a 160-bit hash. In AH, to ensure proper byte boundaries for efficient processing, the authenticating hash is truncated to 96 bits. Expert cryptographers have ascertained that truncating the hash does not lessen its uniqueness or the properties that ensure cryptographic safety. Once the hash has been placed in the Authentication field, along with any other data required by the specific hash algorithm, the message is ready to be sent on its way.

Table 2.1
Classes of IP Header Fields

	IPv4	IPv6
Immutable	Version	Version
	Internet header length	Payload length
	Total length	Next header (AH)
	Identification	Source address
	Protocol (should be value for AH)	Destination address (without routing extension header)
	Source address	—
	Destination address (without source routing)	Destination and hop-by-hop extension headers option type/data length
	Option type/data length/data (classified as immutable)	Destination and hop-by-hop extension headersoption data (option type classified as immutable)
Mutable but Predictable	Destination address (with source routing)	Destination address (with routing extension header)
	—	Routing extension header
Mutable Unpredictable	TOS	Class
	Flags	Flow label
	Fragment offset	Hop limit
	TTL	Destination and hop-by-hop extension headers: option data (option type classified as mutable)
	Header checksum	—
	Option type/data length/data (classified as mutable)	—

8. Fragment the message, if necessary. If the message, enlarged by the AH and possibly by an additional IP header for Tunnel Mode, is sufficiently large that it needs to be fragmented before it is sent, fragmentation takes place at this point. In Transport Mode, the message's source address is always the initiator of the message, so the total message can be authenticated before fragmentation occurs. In Tunnel Mode, the source address of the original header

is the actual initiator of the message; if that source address differs from the outer header's source address, the message may already have been fragmented after it exited the original host. In that case, the tunnel header's authentication was performed on a message fragment, which at this point may have to be further fragmented. Figure 2.7 illustrates the Transport Mode case and Figure 2.8 the Tunnel Mode case.

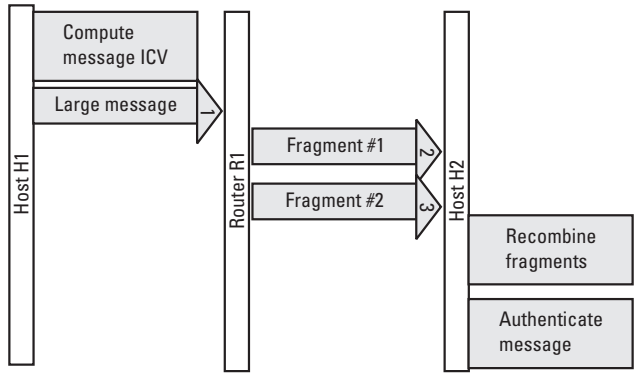


Figure 2.7 Fragmentation and authentication: Transport Mode host-to-host SA.

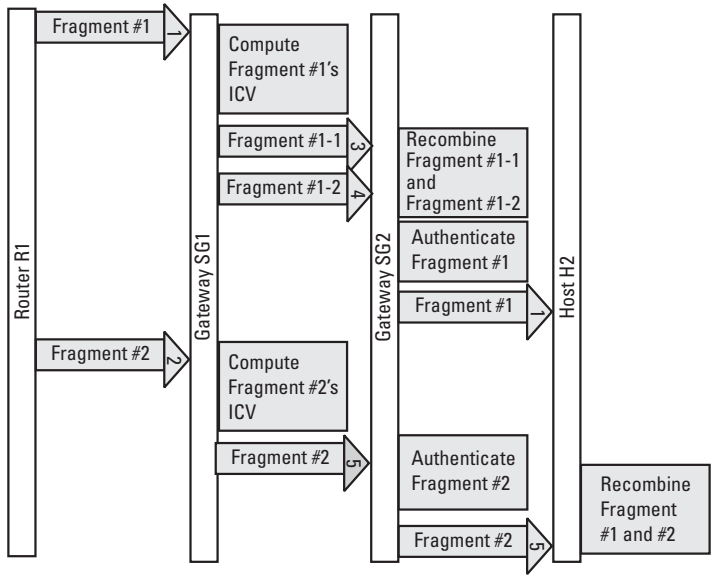


Figure 2.8 Fragmentation and authentication: Tunnel Mode gateway-to-gateway SA.

2.9 AH Processing for Inbound Messages

When a message is received that contains an AH, the IP processing routines first ensure that all fragments of the message have been received and reintegrated to form a complete message. The routines also ensure that the fields that identified each piece of the message as a fragment are reinitialized: The offset field is reset to zero and the "more fragments" flag is turned off, so the IPsec processing routines do not erroneously identify the reassembled message as a message fragment. The message is then passed to the IPsec processing routines, which perform the following steps.

1. Locate the inbound SA governing this protected communication in the SAD. This step is initially accomplished through the use of the three identifying indices: the SPI, the destination address, and the AH protocol. The SA's indices are compared to those found in the packet's outmost AH, whether it is Tunnel Mode or Transport Mode. The packet must also conform to any other selectors that limit the SA's applicability (e.g., port or protocol). If this is a tunnel header, the SA selectors are compared to those found in the packet's inner header, because these fields are not copied into the tunnel header. Once a matching SA has been found, processing can continue. If no such SA is found, the packet is dropped.
2. If replay protection is enabled, perform the replay protection check. The originator of a packet with AH will always increment the replay protection counter; the recipient is free to either ignore this counter or use it to ensure replay protection. However, because IP does not guarantee delivery of packets in the same order in which they were sent (that is the responsibility of the Transport Protocol or the application), this counter cannot be used to ensure exact ordering of the packets, but only a relatively correct order within a window that is a multiple of 32.

   For each inbound SA, the SAD includes a replay window. The size of the window determines how greatly out of order a message can be without being rejected; the size is a multiple of 32, with 64 recommended as a default. A replay window of size N keeps track of the sequence numbers of the last N messages received. Any message with a sequence number so low that it is outside the window's range is dropped. A message within the window's range whose sequence number is a duplicate of a message that was already received is also dropped.

A bit mask (or some equivalent structure) can be used to track the sequence numbers of the last *N* messages received for this SA. Initially, a 64-bit mask could keep track of the receipt of messages with sequence numbers between 1 and 64. Once a message with a sequence number greater than 64 (e.g., sequence number 70) is received, the bit mask would keep track of messages with sequence numbers from 7 to 70; it would then drop any arriving messages with a sequence number less than 7. This check ensures that each inbound message has not been previously received and that it is not grossly out of order. Figure 2.9 illustrates how the sliding replay window works.

3. Verify the authentication data. The authentication hash is computed, in exactly the same manner as for an outbound message. If the computed hash does not match the authentication data found in the message, the message is discarded and no further processing takes place.
4. Strip off the AH and repeat the IPsec processing for any remaining IPsec headers. If there are other nested IPsec headers that terminate at the current destination, each successive header must be processed

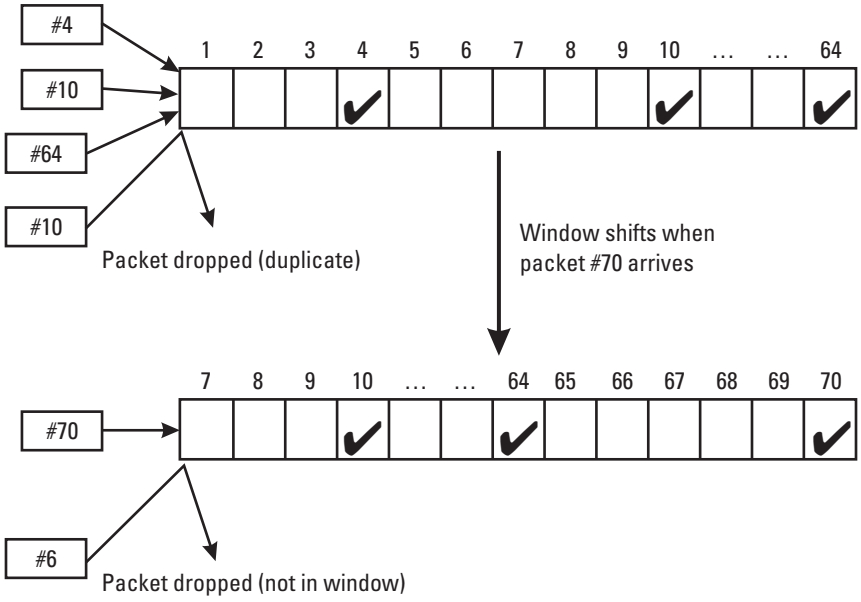


Figure 2.9 The sliding replay protection window.

until one of two conditions is met. Once the last IPsec header is successfully processed, and an upper layer protocol is encountered, the packet is sent to the IP processing routines so it can proceed up the IP stack. Alternatively, if a tunneled IP header is encountered that is not destined for the current host, the packet is forwarded to that destination, where further IPsec processing takes place.

5. Check the SPD to ensure that the IPsec protection applied to the incoming packet conforms to the system's IPsec policy requirements (more on this in Chapter 9). This critical step is difficult to illustrate using only AHs. More impressive examples are possible once we add the other type of security header, the ESP, into the brew in Chapter 3.

2.10 Complications

Two somewhat interrelated aspects of IP networking behavior have the potential to cause severe heartburn for IPsec implementations: packet fragmentation and ICMP [3, 4] error messages.

In scenario 2, let's assume that a Tunnel Mode SA has been established between SG1 and SG2 that protects all traffic between networks N1 and N2. If a packet from host H1-1 to host H2-1 is fragmented before it gets to security gateway SG1 (case 1), either by an intermediate router (IPv4) or by the originating host (IPv6), SG1 computes separate ICVs, one for each fragment. When the fragments reach security gateway SG2, each is authenticated separately, prior to packet reassembly. The reassembled, authenticated packet is then forwarded to its final destination, H2-1. Now let's assume that the packet fragmentation is performed by an intermediate router that lies between SG1 and SG2 (case 2, IPv4 only). SG1 has already computed an ICV for the whole packet. When the fragments reach SG2, they must be reassembled before the packet can be authenticated, because the ICV was computed before fragmentation occurred.

Now let's change the scenario slightly. Assume that SG2, knowing that some segments of the path contain bottlenecks in terms of packet size, decides to do away with the Tunnel Mode SA, thus shortening the size of each packet by avoiding the addition of the outer IP header. This approach, although it does not conform to the prescribed IPsec architecture, has at times been adopted by some implementations. Let's also alter the topology slightly. Unknown to SG2, there is another router or security gateway SG3 (perhaps a back door) serving N2, as illustrated in Figure 2.10. If the SAs

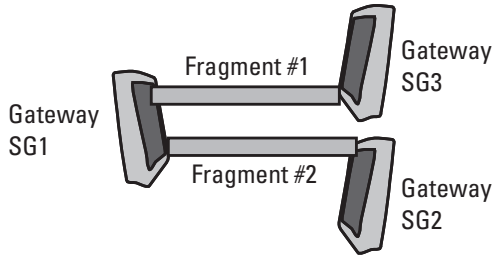


Figure 2.10 Pitfall of (illegal) Transport Mode gateway-to-gateway SAs.

between networks N1 and N2 are all Tunnel Mode SAs, negotiated by SG1 and SG2, all the packet fragments will be routed to the appropriate gateway and the messages properly processed. However, if SG1 and SG2 decide to economize on packet size and establish Transport Mode SAs, problems can ensue. SG2 establishes a Transport Mode header with SG1, under the assumption that it is the only entry point into N2, so that it can grab any protected packets and perform the authentication before the packet reaches H2-1. If any of the fragments are routed via SG3, proper reassembly cannot occur. In case 1, SG2 authenticates each fragment it receives and attempts reassembly. Because all the fragments will not arrive at SG2, the partially reassembled packet is discarded once the reassembly timer expires. Meanwhile, the fragment that arrives at SG3 is either discarded by SG3 or forwarded to H2-1, which, finding no appropriate SA for the fragment, discards it. In case 2, SG2 attempts to reassemble the packet before performing authentication, but otherwise the results are the same as for case 1. This is definitely a worst-case scenario, but in networking worst-case scenarios seem to occur with alarming frequency.

These cases illustrate why the IPsec security architecture requires Tunnel Mode SAs between two gateways, if the SAs protect traffic between hosts other than the two gateways themselves. This also applies to a gateway-to-host SA, in which the gateway protects traffic for other hosts behind the gateway. They also show the complications that fragmentation can cause in the IPsec context.

To avoid fragmentation, gateways must communicate to their protected hosts the size of the headers that the gateway will add to packets sent by the hosts. The originating host generally attempts to send packets that are as close as possible to the PMTU [5–7]. Only by first subtracting the size of the tunnel headers to be added by the security gateway can packet fragmentation be avoided.

There is another way to avoid fragmented packets: The source host can probe the network to ascertain the maximum PMTU for the packet and then adjust the packet size accordingly. In IPv4, this technique also requires that the source host turn on the DF bit, to prevent fragmentation by intermediate routers. This approach can also present problems within the context of IPsec. If a packet is too large to traverse the entire route, an intermediate router sends the ICMP message "packet too big" back to the originating host. In the case of a Tunnel Mode SA, the message is sent back to the security gateway that is the source address of the outer header. It is also significant that the ICMP "packet too big" message used to convey the maximum transmittable packet size (the PMTU) is sent to the packet's source not from the packet's ultimate destination but from an intermediate router. This fact can be very important in an IPsec context, in which we may want to accept only authenticated messages. The gateway then has a problem: Should it believe this unauthenticated message? If it chooses to accept the message as valid, it then has to communicate the message, along with the new PMTU (if included) to the packet's originating host, the source address of the inner header. If the gateway chooses not to relay the message to the host, a black hole situation can occur: The host keeps resending packets with the DF bit on; because it never receives a PMTU message, it does not reduce the packet size. Thus, the packets are continuously resent, adding to network congestion, but they never arrive at their final destination.

The same ICMP messages used to relieve network congestion through the elimination of packet fragmentation can also be used to mount a denial-of-service attack on the network. An attacker can send bogus PMTU messages, with a smaller-than-necessary PMTU. If the gateway accepts unauthenticated PMTU messages and passes them on to the originating host, the host will decrease the packet size for all packets traversing that path. That leads to the transmission of an increased number of small packets, an increasing number of computationally expensive IP-related operations, possibly causing network congestion and a degradation of service.

Several proposals have been advanced to handle the PMTU problem. One possible suggestion involves cooperation between SG1 and SG2. SG1 allows fragmented packets from H1-1 to proceed on their way. To ensure that, if H1-1 sets the DF bit in the inner header, SG1 does not set it in the outer header. When SG2 receives the fragmented packets, it sends a PMTU message to SG1, informing SG1 of the largest fragment size that has successfully traversed the path from SG1 to SG2. Because there is an IPsec tunnel between SG1 and SG2, the PMTU message is protected. This solution differs from the standard PMTU message usage, because the PMTU message is

sent after receipt of a fragmented message; the normal PMTU message results from an unsuccessful attempt to forward an unfragmented message. Alternatively, SG2 can save a PMTU as part of each SA and periodically inform SG1 of the latest PMTU value. If H1-1 attempts to send too large a packet, SG1 can communicate the current PMTU to H1-1. As yet, there is no consensus on the solution to this issue.

Another increasingly common complication is the use of network address translation (NAT) boxes [8–13]. A NAT box can be a separate entity or it can be co-resident with a security gateway. NAT is employed in two different situations. The first is a private network, in which the hosts' addresses must be kept secret for the purposes of security and privacy. The second is a network that uses private addresses that may duplicate addresses used elsewhere on the Internet, because the installation was not assigned enough unique addresses to cover every host. In such a case, a pool of public, globally unique addresses is used for communications with destinations outside the private network. When such messages cross the NAT box, the private source address of an outbound communication is converted to a public address and the public destination address of an inbound communication is converted to the corresponding private network address. That effectively rules out the end-to-end IPsec protection afforded by scenario 1. Because AH authenticates both source and destination addresses, the revised address introduced by the NAT box causes authentication to fail once the message reaches its destination. If the NAT transformations are performed before the IPsec processing for outbound messages and after the IPsec processing for inbound messages, the gateway-to-gateway protection afforded by scenario 2 still is possible. Figure 2.11 shows a workable network configuration incorporating NAT boxes and security gateways.

An IPsec-friendly alternative to NAT, Realm-Specific Internet Protocol (RSIP) [14–16], is emerging. With RSIP, traffic from a host with a private address does not need to use the private addresses for messages intended for destinations outside the private network. The host, acting as an RSIP client, can request a public address from an RSIP server. That way, the message's source address is a globally unique, public address that can be used for end-to-end IPsec protection.

2.11 Auditing

The IPsec documents do not mandate auditing of anomalous or erroneous behavior, because auditing is a process internal to one of the peers and does

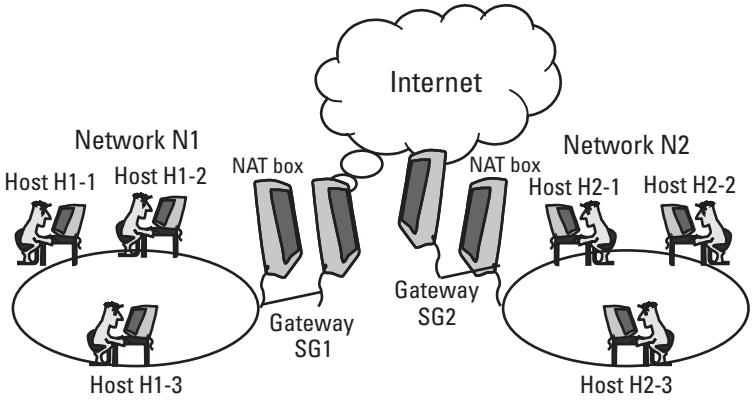


Figure 2.11 Configuring NAT boxes and security gateways.

not change the "bits on the wire." However, events are mentioned that may trigger auditing. If an event is recorded in an audit log, the entry should include the date and time, the source and destination addresses, and the SPI; for IPv6, the flow ID also should be included. In addition, if the system hosting an IPsec implementation does have auditing capabilities, the IPsec implementation is required to support auditing and to allow the system administrator to turn the auditing capability on and off. A warning message is not required to be sent to the peer, because that could start a hailstorm of exchanged messages that could lead to denial of service on one or both machines.

Among the events that can trigger an audit log entry are:

- An attempt to use an outbound SA whose replay counter has reached its maximum value to a recipient that has enabled replay protection;
- An attempt to perform inbound IPsec processing on a message fragment;
- Receipt of an inbound message for which no current, applicable SA can be found;
- Receipt of an inbound message for which verification of the authentication data fails.

In each of those cases, the message is discarded and no further IPsec processing occurs for the discarded message.

2.12 Threat Mitigation

What real-life threats [17, 18] are prevented through the use of the AH? Unauthorized packet alteration can take several forms. The packet content can be altered. The source address can be altered so that the packet appears to come from a sender other than the actual sender; this is called "address spoofing." The packet destination can be altered, in effect rerouting a packet to an unintended recipient. An end-to-end AH, which protects the packet's data, source address, and destination address, protects a packet from all those unauthorized alterations. Unfortunately, if the AH protection is not end to end, and an "unfriendly" user is present on the same network as the source host, that user can capture and alter packets before they reach the gateway that performs the outbound AH processing. Even if the destination address is not altered, a packet can be effectively rerouted if a bogus, unauthenticated DNS message reassigns the destination address (e.g., charlie.org) to the numeric address (e.g., 1.2.3.4) of another host. DNS spoofing can be avoided by accepting only authenticated DNS messages.

AH's replay protection feature can be used to prevent delivery of grossly out-of-order packets, stemming from network problems, an attacker attempting repeated delivery of a significant message (e.g., an electronic funds transfer), or disruption of service via network flooding. The effects of an attacker attempting to bring down a host by flooding it with messages that require expensive cryptographic processing can be mitigated through the use of replay protection, because duplicate packets are discarded before the inbound AH processing takes place.

However, AH does not provide privacy. Even if the packets safely traverse the Internet and arrive intact at their destination, the packets can be read by any of the intermediate nodes that forward the packet on its way. In particular, an attacker can exploit the source routing header option to divert a packet and route it past an evil, information-gathering router. The router can then restore the source routing header to its original form, and the tampering will not be noticed by the recipient.

2.13 Summary

The AH provides several types of critical protection at the network layer. It ensures that messages traversing the Internet arrive at their destination unchanged, that the apparent sender of the message is in actuality the message's originator, and that messages are not erroneously or fraudulently

retransmitted. However, AH does not provide confidentiality to its protected messages. That is the function of the other security-related header, the ESP header.

2.14 Further Reading

AH is definitively described in RFC 2402 [1]. The generalized IPsec architecture, of which AH is an integral part, is defined in RFC 2401 [2]. Two excellent books [17, 18] describe the nature of various security threats and solutions, as well as general security-related information. ICMP for IPv4 is defined in RFC 792 [4]; ICMP for IPv6 is defined in RFC 2463 [3]. The PMTU protocol for IPv4 is described in RFC 1191 [6]; PMTU for IPv6 in RFC 1981 [5]. The interaction of PMTU and security gateways is explored in [7]. NAT is a hotly debated and much analyzed topic; it is defined in RFC 2663 [10] and [12]. The interaction between NAT and IPsec is discussed in [8] and [13]; the interactions between NAT and other protocols are discussed in [19]. An approach to enable NAT to coexist with Tunnel Mode IPsec is defined in [11]. The IAB has issued a report [9] that analyzes NAT's relationship to the Internet's generalized infrastructure and offers guidance on minimizing its negative impact on Internet communications. RSIP is defined in [14] and [15], and its relationship to IPsec is described in [16]. The IPsec email list archive can be found at http://www.vpnc.org/ietf-ipsec.

References

[1] Kent, S., and R. Atkinson, *IP Authentication Header*, RFC 2402, Nov. 1998.

[2] Kent, S., and R. Atkinson, *Security Architecture for the Internet Protocol*, RFC 2401, Nov. 1998.

[3] Conta, A., and S. Deering, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, RFC 2463, Dec. 1998.

[4] Postel, J., *Internet Control Message Protocol*, RFC 792, Sept. 1981.

[5] McCann, J., S. Deering, and J. Mogul, *Path MTU Discovery for IP Version 6*, RFC 1981, Aug. 1996.

[6] Mogul, J., and S. Deering, *Path MTU Discovery*, RFC 1191, Nov. 1990.

[7] Richardson, M., "Path MTU discovery in the presence of security gateways," <draft-richardson-ipsec-pmtu-discov-02.txt>, Aug. 1998.

[8] Aboba, B., "NAT and IPsec," <draft-aboba-nat-ipsec-02.txt>, July 2000.

[9] Hain, T., "Architectural Implications of NAT," <draft-iab-nat-implications-09.txt>, Aug. 2000.

[10] Srisuresh, P., and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations*, RFC 2663, Aug. 1999.

[11] Srisuresh, P., *Security Model With Tunnel-mode IPsec for NAT Domains*, RFC 2709, Oct. 1999.

[12] Srisuresh, P., and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," <draft-ietf-nat-traditional-05.txt>, Oct. 2000.

[13] Stenberg, M., et al., "IPsec NAT Traversal," <draft-stenberg-ipsec-nat-traversal-00.txt>, July 2000.

[14] Borella, M., et al., "Realm Specific IP: Framework," <draft-ietf-nat-rsip-framework-05.txt>, July 2000.

[15] Borella, M., et al., "Realm Specific IP: Protocol Specification," <draft-ietf-nat-rsip-protocol-07.txt>, July 2000.

[16] Montenegro, G., and M. Borella, "RSIP Support for End-to-End IPSEC," <draft-ietf-nat-rsip-ipsec-04.txt>, July 2000.

[17] Cheswick, W. R., and S. M. Bellovin, *Firewalls and Internet Security: Repelling the Wily Hacker*, 2nd Ed., Reading, MA: Addison-Wesley, 2000.

[18] Kaufman, C., R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*, Englewood Cliffs, NJ: Prentice Hall, 1995.

[19] Holdrege, M., and P. Srisuresh, "Protocol Complications With the IP Network Address Translator (NAT)," <draft-ietf-nat-protocol-complications-06.txt>, Oct. 2000.