8.6.1 IPsec

IETF has known for years that security was lacking in the Internet. Adding it was not easy because a war broke out about where to put it. Most security experts believe that to be really secure, encryption and integrity checks have to be end to end (i.e., in the application layer). That is, the source process encrypts and/or integrity protects the data and sends them to the destination process where they are decrypted and/or verified. Any tampering done in between these two processes, including within either operating system, can then be detected. The trouble with this approach is that it requires changing all the applications to make them security aware. In this view, the next best approach is putting encryption in the transport layer or in a new layer between the application layer and the transport layer, making it still end to end but not requiring applications to be changed.

The opposite view is that users do not understand security and will not be capable of using it correctly and nobody wants to modify existing programs in any way, so the network layer should authenticate and/or encrypt packets without the users being involved. After years of pitched battles, this view won enough support that a network layer security standard was defined. In part, the argument was that having network layer encryption does not prevent security-aware users from doing it right and it does help security-unaware users to some extent.

The result of this war was a design called **IPsec** (**IP security**), which is described in RFCs 2401, 2402, and 2406, among others. Not all users want encryption (because it is computationally expensive). Rather than make it optional, it was decided to require encryption all the time but permit the use of a null algorithm. The null algorithm is described and praised for its simplicity, ease of implementation, and great speed in RFC 2410.

The complete IPsec design is a framework for multiple services, algorithms, and granularities. The reason for multiple services is that not everyone wants to pay the price for having all the services all the time, so the services are available a la carte. The major services are secrecy, data integrity, and protection from replay attacks (where the intruder replays a conversation). All of these are based on symmetric-key cryptography because high performance is crucial.

The reason for having multiple algorithms is that an algorithm that is now thought to be secure may be broken in the future. By making IPsec algorithm-independent, the framework can survive even if some particular algorithm is later broken.

The reason for having multiple granularities is to make it possible to protect a single TCP connection, all traffic between a pair of hosts, or all traffic between a pair of secure routers, among other possibilities.

One slightly surprising aspect of IPsec is that even though it is in the IP layer, it is connection oriented. Actually, that is not so surprising because to have any security, a key must be established and used for some period of time—in essence, a kind of connection by a different name. Also, connections amortize the setup

costs over many packets. A "connection" in the context of IPsec is called an **SA** (**Security Association**). An SA is a simplex connection between two endpoints and has a security identifier associated with it. If secure traffic is needed in both directions, two security associations are required. Security identifiers are carried in packets traveling on these secure connections and are used to look up keys and other relevant information when a secure packet arrives.

Technically, IPsec has two principal parts. The first part describes two new headers that can be added to packets to carry the security identifier, integrity control data, and other information. The other part, **ISAKMP** (**Internet Security Association and Key Management Protocol**), deals with establishing keys. ISAKMP is a framework. The main protocol for carrying out the work is **IKE** (**Internet Key Exchange**). Version 2 of IKE as described in RFC 4306 should be used, as the earlier version was deeply flawed, as pointed out by Perlman and Kaufman (2000).

IPsec can be used in either of two modes. In **transport mode**, the IPsec header is inserted just after the IP header. The *Protocol* field in the IP header is changed to indicate that an IPsec header follows the normal IP header (before the TCP header). The IPsec header contains security information, primarily the SA identifier, a new sequence number, and possibly an integrity check of the payload.

In **tunnel mode**, the entire IP packet, header and all, is encapsulated in the body of a new IP packet with a completely new IP header. Tunnel mode is useful when the tunnel ends at a location other than the final destination. In some cases, the end of the tunnel is a security gateway machine, for example, a company firewall. This is commonly the case for a VPN (Virtual Private Network). In this mode, the security gateway encapsulates and decapsulates packets as they pass through it. By terminating the tunnel at this secure machine, the machines on the company LAN do not have to be aware of IPsec. Only the security gateway has to know about it.

Tunnel mode is also useful when a bundle of TCP connections is aggregated and handled as one encrypted stream because it prevents an intruder from seeing who is sending how many packets to whom. Sometimes just knowing how much traffic is going where is valuable information. For example, if during a military crisis, the amount of traffic flowing between the Pentagon and the White House were to drop sharply, but the amount of traffic between the Pentagon and some military installation deep in the Colorado Rocky Mountains were to increase by the same amount, an intruder might be able to deduce some useful information from these data. Studying the flow patterns of packets, even if they are encrypted, is called **traffic analysis**. Tunnel mode provides a way to foil it to some extent. The disadvantage of tunnel mode is that it adds an extra IP header, thus increasing packet size substantially. In contrast, transport mode does not affect packet size as much.

The first new header is **AH** (**Authentication Header**). It provides integrity checking and antireplay security, but not secrecy (i.e., no data encryption). The

use of AH in transport mode is illustrated in Fig. 8-27. In IPv4, it is interposed between the IP header (including any options) and the TCP header. In IPv6, it is just another extension header and is treated as such. In fact, the format is close to that of a standard IPv6 extension header. The payload may have to be padded out to some particular length for the authentication algorithm, as shown.

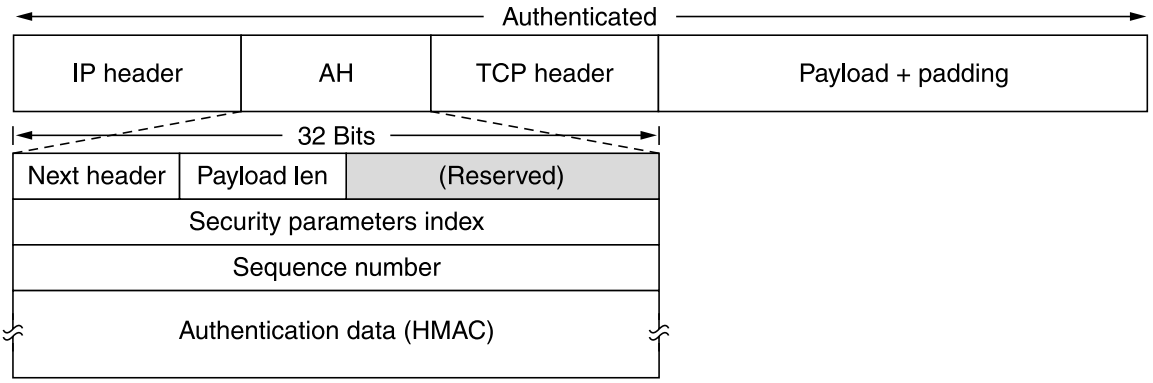

Figure 8-27. The IPsec authentication header in transport mode for IPv4.

Let us now examine the AH header. The *Next header* field is used to store the value that the IP *Protocol* field had before it was replaced with 51 to indicate that an AH header follows. In most cases, the code for TCP (6) will go here. The *Payload length* is the number of 32-bit words in the AH header minus 2.

The *Security parameters index* is the connection identifier. It is inserted by the sender to indicate a particular record in the receiver's database. This record contains the shared key used on this connection and other information about the connection. If this protocol had been invented by ITU rather than IETF, this field would have been called *Virtual circuit number*.

The *Sequence number* field is used to number all the packets sent on an SA. Every packet gets a unique number, even retransmissions. In other words, the retransmission of a packet gets a different number here than the original (even though its TCP sequence number is the same). The purpose of this field is to detect replay attacks. These sequence numbers may not wrap around. If all 2^{32} are exhausted, a new SA must be established to continue communication.

Finally, we come to *Authentication data*, which is a variable-length field that contains the payload's digital signature. When the SA is established, the two sides negotiate which signature algorithm they are going to use. Normally, public-key cryptography is not used here because packets must be processed extremely rapidly and all known public-key algorithms are too slow. Since IPsec is based on symmetric-key cryptography and the sender and receiver negotiate a shared key before setting up an SA, the shared key is used in the signature computation. One simple way is to compute the hash over the packet plus the shared key. The shared key is not transmitted, of course. A scheme like this is called an **HMAC**

(**Hashed Message Authentication Code**). It is much faster to compute than first running SHA-1 and then running RSA on the result.

The AH header does not allow encryption of the data, so it is mostly useful when integrity checking is needed but secrecy is not needed. One noteworthy feature of AH is that the integrity check covers some of the fields in the IP header, namely, those that do not change as the packet moves from router to router. The *Time to live* field changes on each hop, for example, so it cannot be included in the integrity check. However, the IP source address is included in the check, making it impossible for an intruder to falsify the origin of a packet.

The alternative IPsec header is **ESP** (**Encapsulating Security Payload**). Its use for both transport mode and tunnel mode is shown in Fig. 8-28.

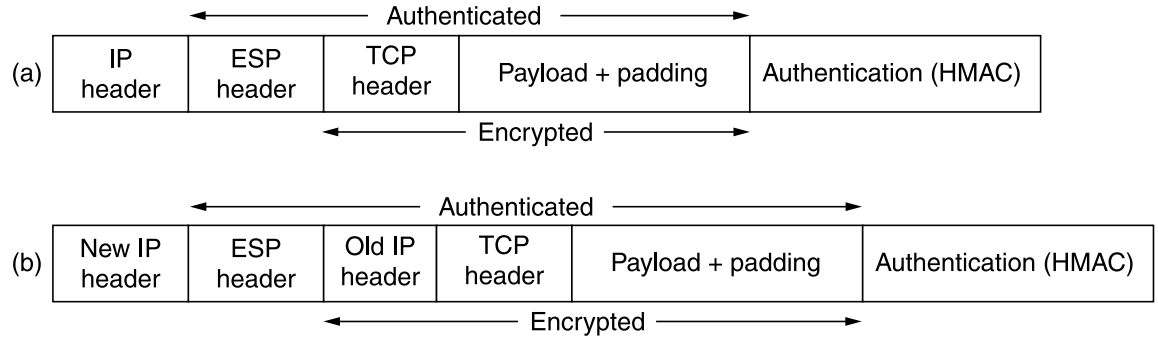

Figure 8-28. (a) ESP in transport mode. (b) ESP in tunnel mode.

The ESP header consists of two 32-bit words. They are the *Security parameters index* and *Sequence number* fields that we saw in AH. A third word that generally follows them (but is technically not part of the header) is the *Initialization vector* used for the data encryption, unless null encryption is used, in which case it is omitted.

ESP also provides for HMAC integrity checks, as does AH, but rather than being included in the header, they come after the payload, as shown in Fig. 8-28. Putting the HMAC at the end has an advantage in a hardware implementation: the HMAC can be calculated as the bits are going out over the network interface and appended to the end. This is why Ethernet and other LANs have their CRCs in a trailer, rather than in a header. With AH, the packet has to be buffered and the signature computed before the packet can be sent, potentially reducing the number of packets/sec that can be sent.

Given that ESP can do everything AH can do and more and is more efficient to boot, the question arises: why bother having AH at all? The answer is mostly historical. Originally, AH handled only integrity and ESP handled only secrecy. Later, integrity was added to ESP, but the people who designed AH did not want to let it die after all that work. Their only real argument is that AH checks part of the IP header, which ESP does not, but other than that it is really a weak argument. Another weak argument is that a product supporting AH but not ESP might