PART SIX

Internet Applications

Part Six looks at a range of applications that operate over the Internet.

ROAD MAP FOR PART SIX

Chapter 21 Network Security

Network security has become increasingly important with the growth in the number and importance of networks. Chapter 21 provides a survey of security techniques and services. The chapter begins with a look at encryption techniques for ensuring confidentiality, which include the use of conventional and public-key encryption. Then the area of authentication and digital signatures is explored. The two most important encryption algorithms, AES and RSA, are examined, as well as SHA-1, a one-way hash function important in a number of security applications. Chapter 21 also discusses SSL and the set of IP security standards.

Chapter 22 Internet Applications—Electronic Mail and Network Management

The purpose of a communications architecture is to support distributed applications. Chapter 22 examines two of the most important of these applications; in each case, general principles are discussed, followed by a specific example. The applications discussed are electronic mail and network management. The corresponding examples are SMTP and MIME; and SNMP.

Chapter 23 Internet Applications—Internet Directory Service and World Wide Web

Chapter 23 looks at several important modern application areas for the Internet. The Domain Name System (DNS) is a directory lookup service that provides a mapping between the name of a host on the Internet and its numerical address; it is a required application as specified in RFC 1123. The Hypertext Transfer Protocol (HTTP) supports the exchange of requests and responses between Web browsers and Web servers.

Chapter 24 Internet Applications—Multimedia

Chapter 24 examines key topics related to multimedia. The chapter begins with a discussion of audio and video compression. The Session Initiation Protocol (SIP) is an application-level control protocol for setting up, modifying, and terminating real-time sessions between participants over an IP data network; these include telephony and multimedia sessions. Finally, this chapter examines the Real-Time Transport Protocol (RTP).

CHAPTER 21

NETWORK SECURITY

21.1 Security Requirements and Attacks

21.2 Confidentiality with Symmetric Encryption

21.3 Message Authentication and Hash Functions

21.4 Public-Key Encryption and Digital Signatures

21.5 Secure Socket Layer and Transport Layer Security

21.6 IPv4 and IPv6 Security

21.7 Wi-Fi Protected Access

21.8 Recommended Reading and Web Sites

21.9 Key Terms, Review Questions, and Problems

To guard against the baneful influence exerted by strangers is therefore an elementary dictate of savage prudence. Hence before strangers are allowed to enter a district, or at least before they are permitted to mingle freely with the inhabitants, certain ceremonies are often performed by the natives of the country for the purpose of disarming the strangers of their magical powers, or of disinfecting, so to speak, the tainted atmosphere by which they are supposed to be surrounded.

—*The Golden Bough*, Sir James George Frazer

KEY POINTS

- Network security threats fall into two categories. **Passive threats**, sometimes referred to as eavesdropping, involve attempts by an attacker to obtain information relating to a communication. **Active threats** involve some modification of the transmitted data or the creation of false transmissions.
- By far the most important automated tool for network and communications security is **encryption**. With **symmetric encryption**, two parties share a single encryption/decryption key. The principal challenge with symmetric encryption is the distribution and protection of the keys. A **public-key encryption** scheme involves two keys, one for encryption and a paired key for decryption. The party that generated the key pair keeps one of the keys private and makes the other key public.
- Symmetric encryption and public-key encryption are often combined in secure networking applications. Symmetric encryption is used to encrypt transmitted data, using a one-time or short-term session key. The session key can be distributed by a trusted key distribution center or transmitted in encrypted form using public-key encryption. Public-key encryption is also used to create digital signatures, which can authenticate the source of transmitted messages.
- The Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS) provide security services for Web transactions.
- A security enhancement used with both IPv4 and IPv6, called IPSec, provides both confidentiality and authentication mechanisms.

The requirements of **information security** within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An

example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents. An example of the latter is personnel screening procedures used during the hiring process.

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone or data network. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**. Although this is an important topic, it is beyond the scope of this book.

The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. **Network security** measures are needed to protect data during their transmission and to guarantee that data transmissions are authentic.

The essential technology underlying virtually all automated network and computer security applications is encryption. Two fundamental approaches are in use: symmetric encryption and public-key encryption, also known as asymmetric encryption. As we look at the various approaches to network security, these two types of encryption will be explored.

The chapter begins with an overview of the requirements for network security. Next, we look at symmetric encryption and its use to provide confidentiality. This is followed by a discussion of message authentication. We then look at the use of public-key encryption and digital signatures. The chapter closes with an examination of security features in SSL, IPSec, and Wi-Fi Protected Access.

21.1 SECURITY REQUIREMENTS AND ATTACKS

To understand the types of threats to security that exist, we need to have a definition of security requirements. Computer and network security address four requirements:

- **Confidentiality:** Requires that data only be accessible by authorized parties. This type of access includes printing, displaying, and other forms of disclosure, including simply revealing the existence of an object.
- **Integrity:** Requires that only authorized parties can modify data. Modification includes writing, changing, changing status, deleting, and creating.
- **Availability:** Requires that data are available to authorized parties.
- **Authenticity:** Requires that a host or service be able to verify the identity of a user.

A useful means of classifying security attacks (RFC 2828) is in terms of *passive attacks* and *active attacks*. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, or a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

A second type of passive attack, **traffic analysis**, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. Even with encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

A **masquerade** takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect. For example, a message meaning "Allow John Smith to read confidential file *accounts*" is modified to mean "Allow Fred Brown to read confidential file *accounts*."

The **denial of service** prevents or inhibits the normal use or management of communications facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network or a server, either by disabling the network server or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communications facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them. Because the detection has a deterrent effect, it may also contribute to prevention.

21.2 CONFIDENTIALITY WITH SYMMETRIC ENCRYPTION

The universal technique for providing confidentiality for transmitted data is symmetric encryption. This section looks first at the basic concept of symmetric encryption, followed by a discussion of the two most important symmetric encryption algorithms: the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES). We then examine the application of symmetric encryption to achieve confidentiality.

Symmetric Encryption

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the introduction of public-key encryption in the late 1970s. Countless individuals and groups, from Julius Caesar to the German U-boat force to present-day diplomatic, military, and commercial users, have used symmetric encryption for secret communication. It remains by far the more widely used of the two types of encryption.

A symmetric encryption scheme has five ingredients (Figure 21.1):

- **Plaintext:** This is the original message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.

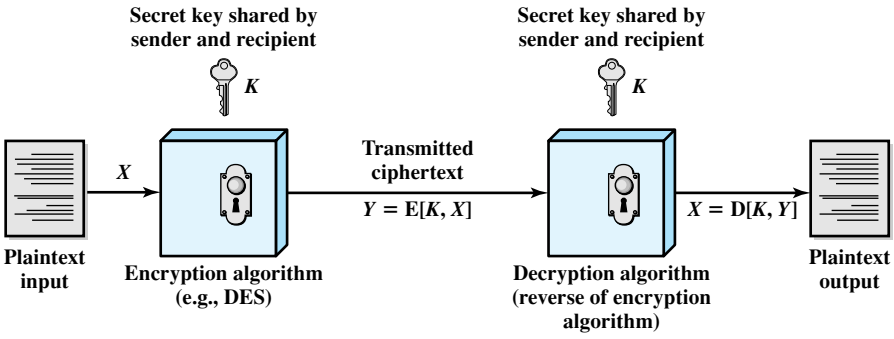

Figure 21.1 Simplified Model of Symmetric Encryption

- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of symmetric encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

There are two general approaches to attacking a symmetric encryption scheme. The first attack is known as **cryptanalysis**. Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

The second method, known as the **brute-force** attack, is to try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success. Table 21.1 shows how much time is involved for various key sizes. The table shows results for each key size, assuming that it takes 1 μs to perform a single decryption, a reasonable order of magnitude for today's computers. With the use of massively parallel organizations of microprocessors, it may be possible to achieve processing rates many orders of magnitude greater. The final column of the table considers the results for a system that can process 1 million keys per microsecond. As one can see, at this performance level, a 56-bit key can no longer be considered computationally secure.

Table 21.1 Average Time Required for Exhaustive Key Search

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	2^{31} μs = 35.8 minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	2^{55} μs = 1142 years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	2^{167} μs = 5.9×10^{36} years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	2×10^{26} μs = 6.4×10^{12} years	6.4×10^6 years

Encryption Algorithms

The most commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks and produces a block of ciphertext of equal size for each plaintext block. The two most important symmetric algorithms, both of which are block ciphers, are the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES).

Data Encryption Standard DES has been the dominant encryption algorithm since its introduction in 1977. However, because DES uses only a 56-bit key, it was only a matter of time before computer processing speed made DES obsolete. In 1998, the Electronic Frontier Foundation (EFF) announced that it had broken a DES challenge using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98]. And, of course, hardware prices will continue to drop as speeds increase, making DES worthless.

The life of DES was extended by the use of triple DES (3DES), which involves repeating the basic DES algorithm three times, using either two or three unique keys, for a key size of 112 or 168 bits.

The principal drawback of 3DES is that the algorithm is relatively sluggish in software. A secondary drawback is that both DES and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

Advanced Encryption Standard Because of these drawbacks, 3DES is not a reasonable candidate for long-term use. As a replacement, the National Institute of Standards and Technology (NIST) in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which should have a security strength equal to or better than 3DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. Evaluation criteria include security, computational efficiency, memory requirements, hardware and software suitability, and flexibility. In 2001, AES was issued as a federal information processing standard (FIPS 197).

In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented.

Figure 21.2 shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS 197, this block is depicted as a square matrix of bytes. This block is copied into the **State** array, which is modified at each stage of encryption or decryption. After the final stage, **State** is copied to an output matrix. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is 4 bytes and the total key schedule is 44 words for the 128-bit key. The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the **in** matrix, the second four bytes occupy the second column, and so on.

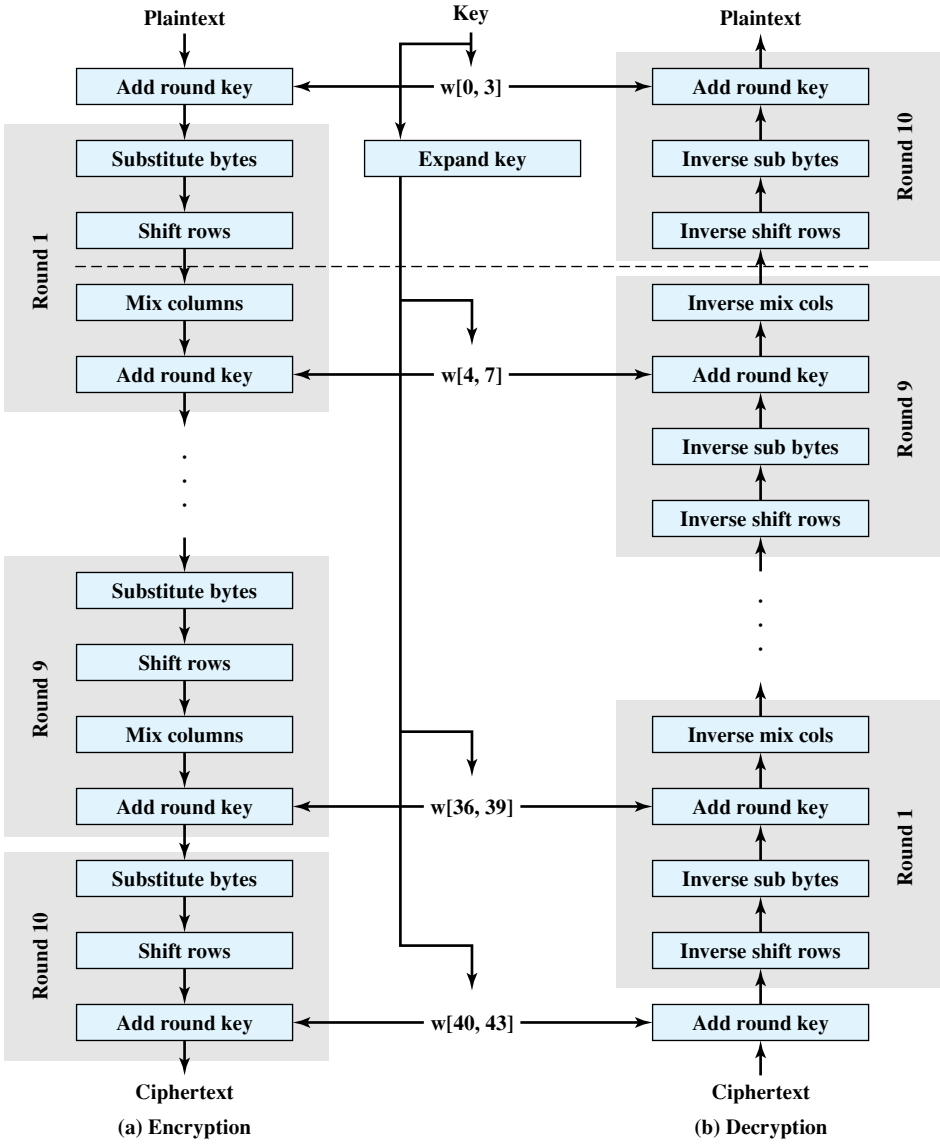

Figure 21.2 AES Encryption and Decryption

Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the **w** matrix.

The following comments give some insight into AES:

1. The key that is provided as input is expanded into an array of forty-four 32-bit words, **w**[*i*]. Four distinct words (128 bits) serve as a round key for each round.
2. Four different stages are used, one of permutation and three of substitution:

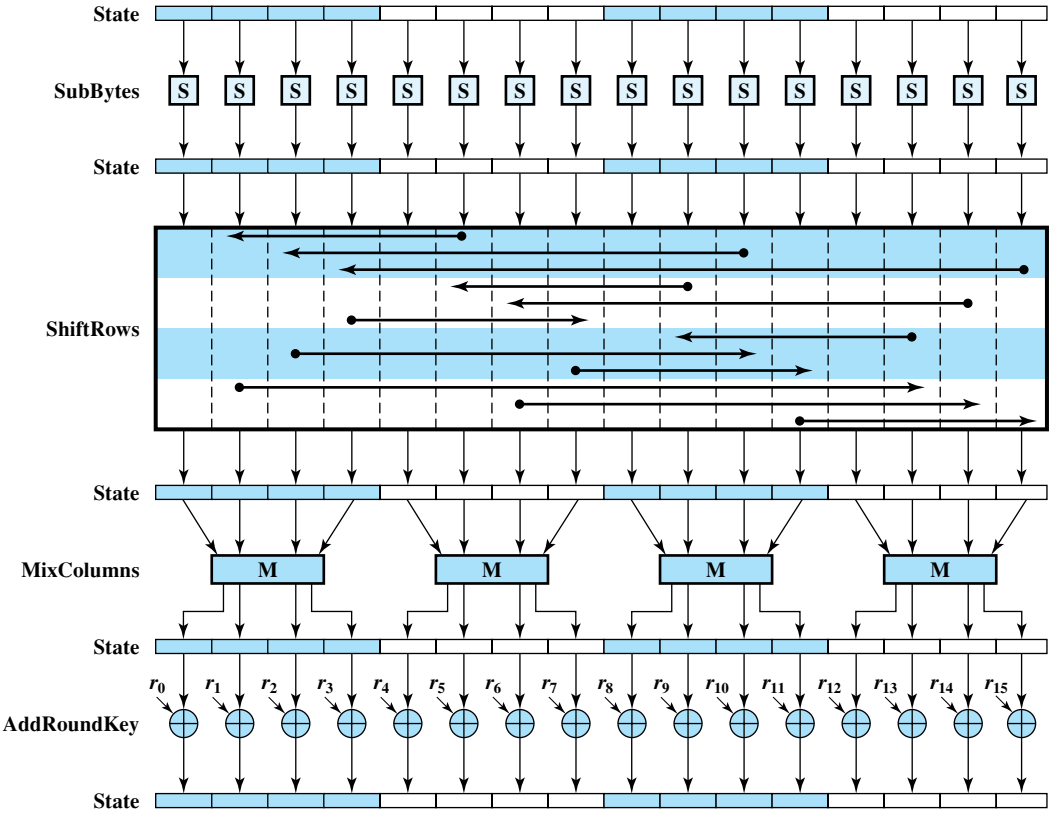

Figure 21.3 AES Encryption Round

- **Substitute bytes:** Uses a table, referred to as an S-box,[1] to perform a byte-by-byte substitution of the block
- **Shift rows:** A simple permutation that is performed row by row
- **Mix columns:** A substitution that alters each byte in a column as a function of all of the bytes in the column
- **Add round key:** A simple bitwise XOR of the current block with a portion of the expanded key

3. The structure is quite simple. For both encryption and decryption, the cipher begins with an Add Round Key stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. Figure 21.3 depicts the structure of a full encryption round.
4. Only the Add Round Key stage makes use of the key. For this reason, the cipher begins and ends with an Add Round Key stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.

[1]The term *S-box*, or substitution box, is commonly used in the description of symmetric ciphers to refer to a table used for a table-lookup type of substitution mechanism.

5. The Add Round Key stage by itself would not be formidable. The other three stages together scramble the bits, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (Add Round Key) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure.
6. Each stage is easily reversible. For the Substitute Byte, Shift Row, and Mix Columns stages, an inverse function is used in the decryption algorithm. For the Add Round Key stage, the inverse is achieved by XORing the same round key to the block, using the result that $A \oplus A \oplus B = B$.
7. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES.
8. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. Figure 21.2 lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g., the dashed line in the figure), **State** is the same for both encryption and decryption.
9. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

Location of Encryption Devices

The most powerful, and most common, approach to countering the threats to network security is encryption. In using encryption, we need to decide what to encrypt and where the encryption gear should be located. As Figure 21.4

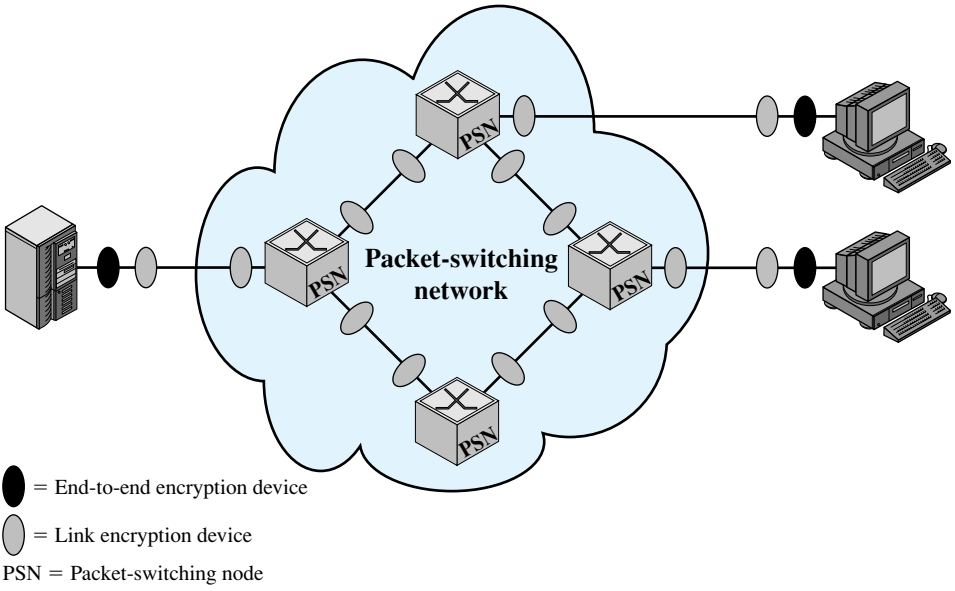

Figure 21.4 Encryption across a Packet-Switching Network

indicates, there are two fundamental alternatives: link encryption and end-to-end encryption.

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. Although this requires a lot of encryption devices in a large network, it provides a high level of security. One disadvantage of this approach is that the message must be decrypted each time it enters a packet switch; this is necessary because the switch must read the address (virtual circuit number) in the packet header to route the packet. Thus, the message is vulnerable at each switch. If this is a public packet-switching network, the user has no control over the security of the nodes.

With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data, in encrypted form, are then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This approach would seem to secure the transmission against attacks on the network links or switches. There is, however, still a weak spot.

Consider the following situation. A host connects to a frame relay network, sets up a logical connection to another host, and is prepared to transfer data to that other host using end-to-end encryption. Data are transmitted over such a network in the form of frames, or packets, consisting of a header and some user data. What part of each packet will the host encrypt? Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryption. Each frame relay node will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet. It follows that the host may only encrypt the user data portion of the packet and must leave the header in the clear, so that the network can read it.

Thus, with end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the clear. To achieve greater security, both link and end-to-end encryption are needed, as is shown in Figure 21.4.

To summarize, when both forms are employed, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key. As the packet traverses the network, each switch decrypts the packet using a link encryption key to read the header and then encrypts the entire packet again for sending it out on the next link. Now the entire packet is secure except for the time that the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

Key Distribution

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a key to two parties that wish to exchange data without allowing others to see the key. Key distribution can be achieved in a number of ways. For two parties A and B,

1. A key could be selected by A and physically delivered to B.
2. A third party could select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key.
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is only going to be exchanging data with its partner on the other end of the link. However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs a number of keys, supplied dynamically. The problem is especially difficult in a wide area distributed system.

Option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys are revealed. Even if frequent changes are made to the link encryption keys, these should be done manually. To provide keys for end-to-end encryption, option 4 is preferable.

Figure 21.5 illustrates an implementation of option 4 for end-to-end encryption. In the figure, link encryption is ignored. This can be added, or not, as required. For this scheme, two kinds of keys are identified:

- **Session key:** When two end systems (hosts, terminals, etc.) wish to communicate, they establish a logical connection (e.g., virtual circuit). For the duration of that logical connection, all user data are encrypted with a one-time session key. At the conclusion of the session, or connection, the session key is destroyed.

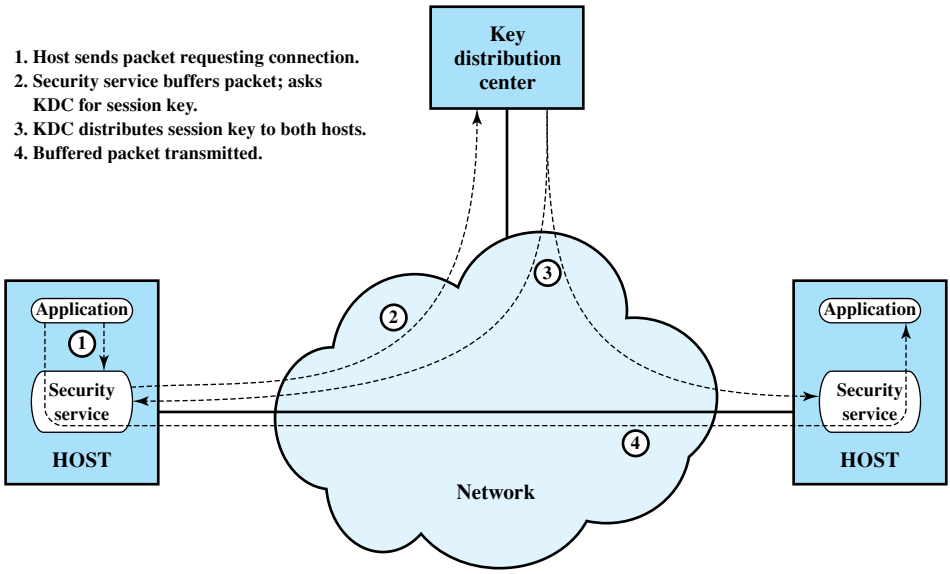

Figure 21.5 Automatic Key Distribution for Connection-Oriented Protocol

- **Permanent key:** A permanent key is a key used between entities for the purpose of distributing session keys.

The configuration consists of the following elements:

- **Key distribution center:** The key distribution center determines which systems are allowed to communicate with each other. When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.
- **Security service module (SSM):** This module, which may consist of functionality at one protocol layer, performs end-to-end encryption and obtains session keys on behalf of users.

The steps involved in establishing a connection are shown in the figure. When one host wishes to set up a connection to another host, it transmits a connection-request packet (step 1). The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2). The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM (step 3). The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4). All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.

The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

Another approach to key distribution uses public-key encryption, which is discussed in Section 21.4.

Traffic Padding

We mentioned that, in some cases, users are concerned about security from traffic analysis. With the use of link encryption, packet headers are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding.

Traffic padding is a function that produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, the random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and noise and therefore impossible to deduce the amount of traffic.

21.3 MESSAGE AUTHENTICATION AND HASH FUNCTIONS

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

Approaches to Message Authentication

A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received messages are authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message's timeliness (it has not been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

Authentication Using Symmetric Encryption It is possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able successfully to encrypt a message for the other participant. Furthermore, if the message includes an error detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

Message Authentication without Message Encryption In this section, we examine several approaches to message authentication that do not rely on message encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. Because symmetric encryption will provide authentication, and because it is widely used with readily available products, why not simply use such an approach, which provides both confidentiality and authentication? [DAVI89] suggests three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations. For example, notification to users that the network is now unavailable or an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.
2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, with messages chosen at random for checking.
3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.

Thus, there is a place for both authentication and encryption in meeting security requirements.

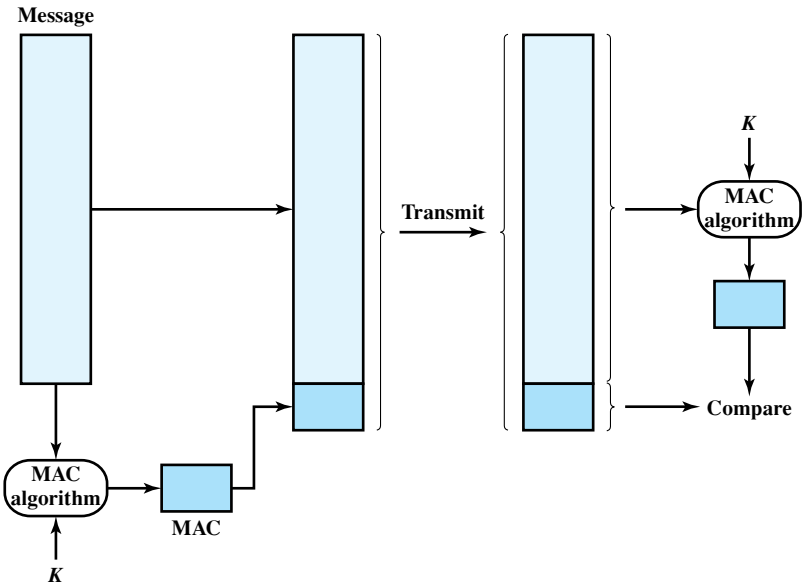

Figure 21.6 Message Authentication Using a Message Authentication Code (MAC)

Message Authentication Code One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K_{AB}. When A has a message M to send to B, it calculates the message authentication code as a function of the message and the key: $\text{MAC}_M = \text{F}(K_{AB}, M)$. The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code (Figure 21.6). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.
3. If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A number of algorithms could be used to generate the code. The National Bureau of Standards, in its publication *DES Modes of Operation*, recommends the

use of DES. DES is used to generate an encrypted version of the message, and the last number of bits of ciphertext are used as the code. A 16- or 32-bit code is typical.

The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption. It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

One-Way Hash Function A variation on the message authentication code that has received much attention recently is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size message digest $H(M)$ as output. Unlike the MAC, a hash function does not also take a secret key as input. To authenticate a message, the message digest is sent with the message in such a way that the message digest is authentic.

Figure 21.7 illustrates three ways in which the message can be authenticated. The message digest can be encrypted using symmetric encryption (part a); if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured. The message digest can also be encrypted using public-key encryption (part b); this is explained in Section 21.4. The public-key approach has two advantages: it provides a digital signature as well as message authentication, and it does not require the distribution of keys to communicating parties.

These two approaches have an advantage over approaches that encrypt the entire message in that less computation is required. Nevertheless, there has been interest in developing a technique that avoids encryption altogether. Several reasons for this interest are pointed out in [TSUD92]:

- Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- Encryption hardware costs are nonnegligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents. Some encryption algorithms, such as the RSA public-key algorithm, are patented and must be licensed, adding a cost.
- Encryption algorithms may be subject to export control.

Figure 21.7c shows a technique that uses a hash function but no encryption for message authentication. This technique assumes that two communicating parties, say A and B, share a common secret value S_{AB}. When A has a message to send to B, it calculates the hash function over the concatenation of the secret value and the message: $MD_M = H(S_{AB} \| M)$.[2] It then sends $[M \| MD_M]$ to B. Because B possesses S_{AB}, it can recompute $H(S_{AB} \| M)$ and verify MD_M. Because the secret value itself is not sent, it is not possible for an attacker to modify an intercepted message. As long

[2] $\|$ denotes concatenation.

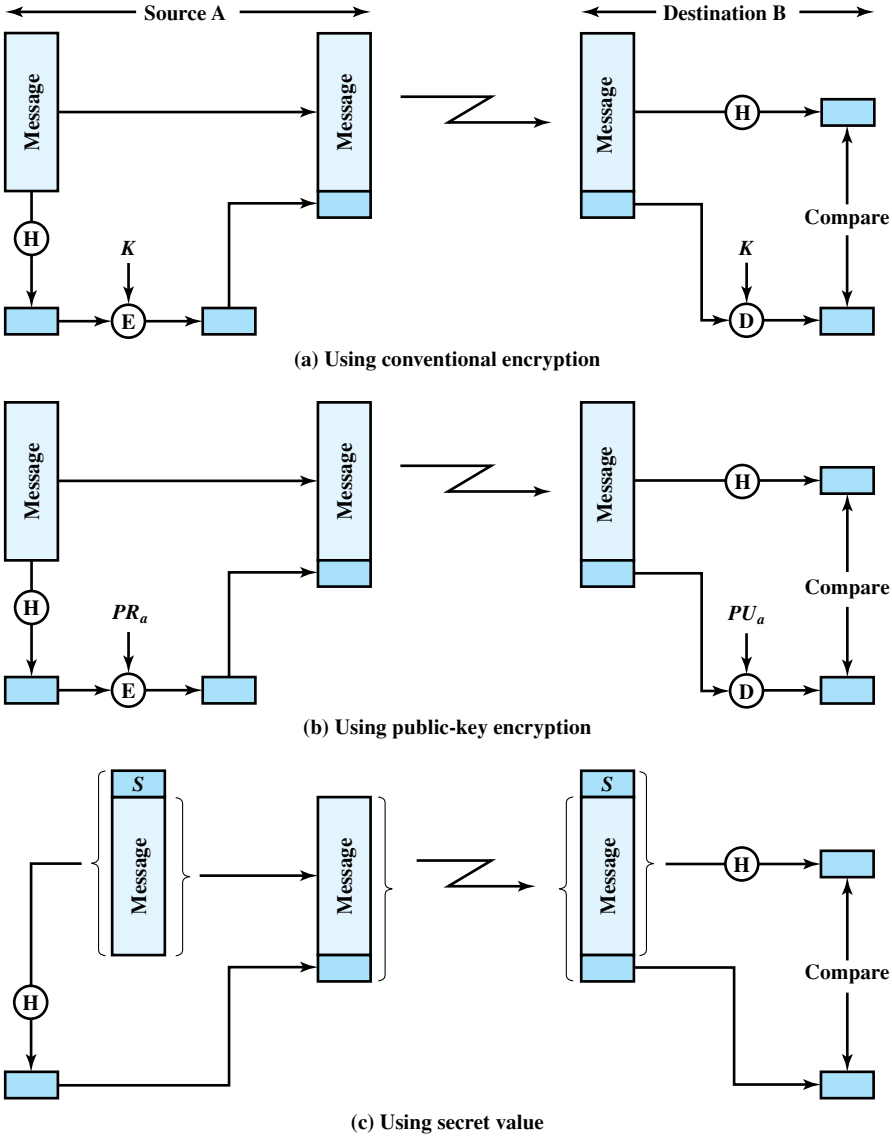

Figure 21.7 Message Authentication Using a One-Way Hash Function

as the secret value remains secret, it is also not possible for an attacker to generate a false message.

This third technique, using a shared secret value, is the one adopted for IP security; it has also been specified for SNMPv3, discussed in Chapter 22.

Secure Hash Functions

The one-way hash function, or secure hash function, is important not only in message authentication but in digital signatures. In this section, we begin with

a discussion of requirements for a secure hash function. Then we look at one of the most important hash functions, SHA.

Hash Function Requirements The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given code h, it is computationally infeasible to find x such that $H(x) = h$.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
6. It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property is the one-way property: It is easy to generate a code given a message, but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value (Figure 21.7c). The secret value itself is not sent; however, if the hash function is not one way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message M and the hash code $MD_M = H(S_{AB} \| M)$. The attacker then inverts the hash function to obtain $S_{AB} \| M = H^{-1}(MD_M)$. Because the attacker now has both M and $S_{AB} \| M$, it is a trivial matter to recover S_{AB}.

The fifth property guarantees that it is impossible to find an alternative message with the same hash value as a given message. This prevents forgery when an encrypted hash code is used (Figures 21.7a and b). If this property were not true, an attacker would be capable of the following sequence: First, observe or intercept a message plus its encrypted hash code; second, generate an unencrypted hash code from the message; third, generate an alternate message with the same hash code.

A hash function that satisfies the first five properties in the preceding list is referred to as a weak hash function. If the sixth property is also satisfied, then it is referred to as a strong hash function. The sixth property protects against a sophisticated class of attack known as the birthday attack.[3]

In addition to providing authentication, a message digest also provides data integrity. It performs the same function as a frame check sequence: If any bits in the message are accidentally altered in transit, the message digest will be in error.

The SHA Secure Hash Function

The Secure Hash Algorithm (SHA) was developed by NIST and published as a federal information processing standard (FIPS 180) in 1993; a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1.

[3]See [STAL06] for a discussion of birthday attacks.

SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a new version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on the other SHA versions by 2010. Shortly thereafter, a research team described an attack in which two separate messages could be found that deliver the same SHA-1 hash using 2^{69} operations, far fewer than the 2^{80} operations previously thought needed to find a collision with an SHA-1 hash [WANG05]. This result should hasten the transition to the other versions of SHA.

In this section, we provide a description of SHA-512. The other versions are quite similar. The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. Figure 21.8 depicts the overall processing of a message to produce a digest. The processing consists of the following steps:

Step 1: Append padding bits. The message is padded so that its length is congruent to 896 modulo 1024 [length mod $1024 = 896$). Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding) The inclusion of a length value makes more difficult a kind of attack known as a padding attack [TSUD92].

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 21.8, the expanded message is represented as the

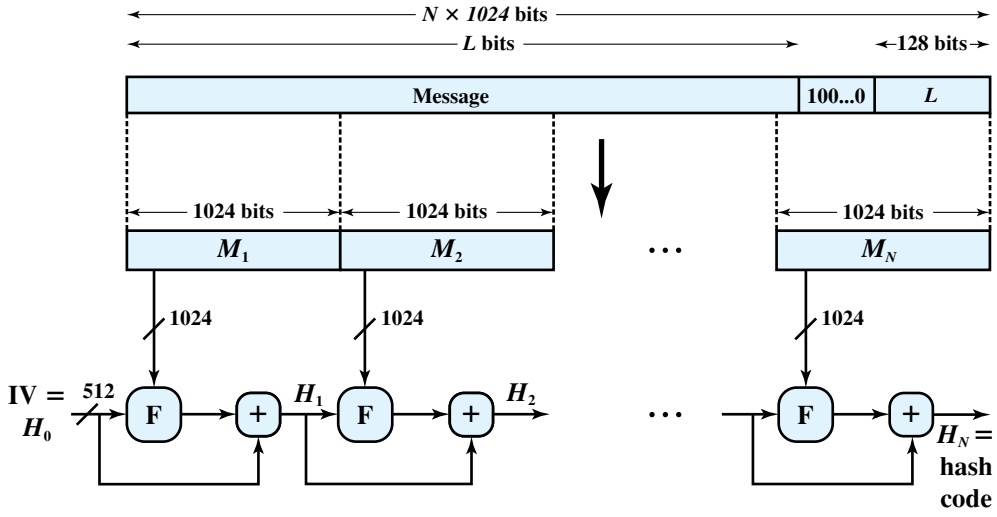

Figure 21.8 Message Digest Generation Using SHA-512

sequence of 1024-bit blocks $M_1, M_2, \ldots, M_N$, so that the total length of the expanded message is $N \times 1024$ bits.

Step 3: Initialize MD buffer. A 512-bit buffer is used to hold intermediate and final results of the hash function.

Step 4: Process message in 512-bit (16-word) blocks. The heart of the algorithm is a module that consists of 80 rounds of processing. The 80 rounds have the same structure but vary some constants and logical functions.

Step 5: Output. After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

The SHA-512 algorithm has the property that every bit of the hash code is a function of every bit of the input. The complex repetition of the basic function F produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code. Unless there is some hidden weakness in SHA-512, which has not so far been published, the difficulty of coming up with two messages having the same message digest is on the order of 2^{256} operations, while the difficulty of finding a message with a given digest is on the order of 2^{512} operations.

21.4 PUBLIC-KEY ENCRYPTION AND DIGITAL SIGNATURES

Of equal importance to symmetric encryption is public-key encryption, which finds use in message authentication and key distribution. This section looks first at the basic concept of public-key encryption, followed by a discussion of digital signatures. Then we discuss the most widely used public-key algorithm: RSA. We then look at the problem of key distribution.

Public-Key Encryption

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976 [DIFF76], is the first truly revolutionary advance in encryption in literally thousands of years. For one thing, public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns. More important, public-key cryptography is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

Before proceeding, we should first mention several common misconceptions concerning public-key encryption. One is that public-key encryption is more secure from cryptanalysis than symmetric encryption. In fact, the security of any encryption scheme depends on (1) the length of the key and (2) the computational work involved in breaking a cipher. There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the viewpoint of resisting cryptanalysis. A second misconception is that public-key encryption is a general-purpose technique that has made symmetric encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be

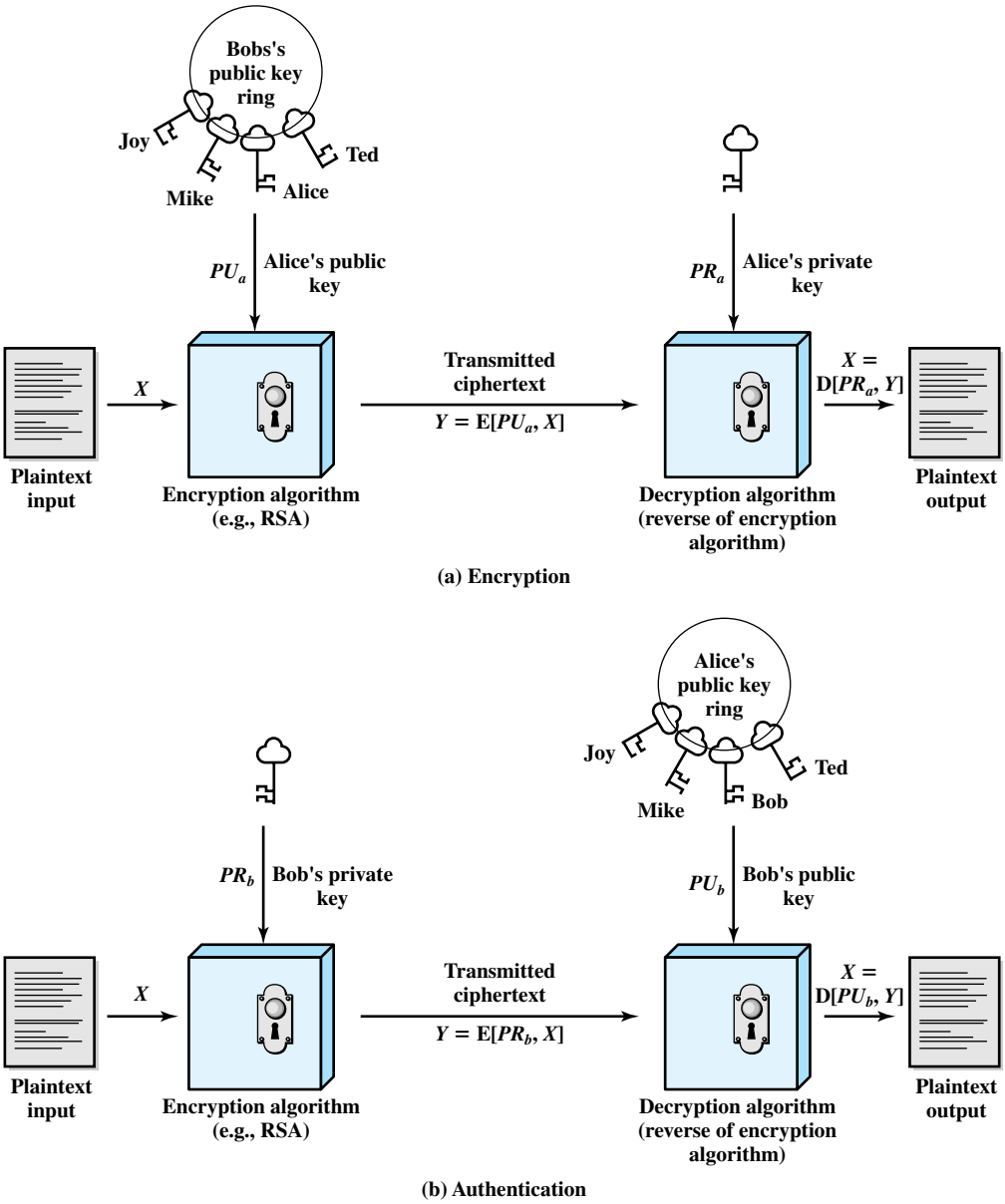

Figure 21.9 Public-Key Cryptography

abandoned. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for symmetric encryption. In fact, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for symmetric encryption.

A public-key encryption scheme has six ingredients (Figure 21.9):

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

As the names suggest, the public key of the pair is made public for others to use, while the private key is known only to its owner. A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption. Furthermore, these algorithms have the following important characteristics:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
- For most public-key schemes, either of the two related keys can be used for encryption, with the other used for decryption.

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 21.9 suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user protects his or her private key, incoming communication is secure. At any time, a user can change the private key and publish the companion public key to replace the old public key.

Digital Signature

Public-key encryption can be used in another way, as illustrated in Figure 21.9b. Suppose that Bob wants to send a message to Alice and, although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. In

this case Bob uses his own private key to encrypt the message. When Alice receives the ciphertext, she finds that she can decrypt it with Bob's public key, thus proving that the message must have been encrypted by Bob. No one else has Bob's private key and therefore no one else could have created a ciphertext that could be decrypted with Bob's public key. Therefore, the entire encrypted message serves as a **digital signature**. In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

In the preceding scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the sender's private key, it serves as a signature that verifies origin, content, and sequencing. A secure hash code such as SHA-1 can serve this function.

It is important to emphasize that the digital signature does not provide confidentiality. That is, the message being sent is safe from alteration but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message, because the rest of the message is transmitted in the clear. Even in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

The RSA Public-Key Encryption Algorithm

One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since that time reigned supreme as the only widely accepted and implemented approach to public-key encryption. RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n.

Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$$C = M^e \bmod n$$
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

Both sender and receiver must know the values of n and e, and only the receiver knows the value of d. This is a public-key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$. For this algorithm to be satisfactory for public-key encryption, the following requirements must be met:

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate M^e and C^d for all values of $M < n$.
3. It is infeasible to determine d given e and n.

The first two requirements are easily met. The third requirement can be met for large values of e and n.

Figure 21.10 summarizes the RSA algorithm. Begin by selecting two prime numbers, p and q, and calculating their product, n, which is the modulus for encryption and

Key Generation

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$de \bmod \phi(n) = 1$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption

Plaintext:	$M < n$
Ciphertext:	$C = M^e \pmod n$

Decryption

Ciphertext:	C
Plaintext:	$M = C^d \pmod n$

Figure 21.10 The RSA Algorithm

decryption. Next, we need the quantity $\phi(n)$, referred to as the Euler totient of n, which is the number of positive integers less than n and relatively prime to n.[4] Then select an integer e that is relatively prime to $\phi(n)$ [i.e., the greatest common divisor of e and $\phi(n)$ is 1]. Finally, calculate d such that $de \bmod \phi(n) = 1$. It can be shown that d and e have the desired properties.

Suppose that user A has published its public key and that user B wishes to send the message M to A. Then B calculates $C = M^e \pmod n$ and transmits C. On receipt of this ciphertext, user A decrypts by calculating $M = C^d \pmod n$.

An example, from [SING99], is shown in Figure 21.11. For this example, the keys were generated as follows:

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.

[4]It can be shown that when n is a product of two primes, pq, then $\phi(n) = (p - 1)(q - 1)$.

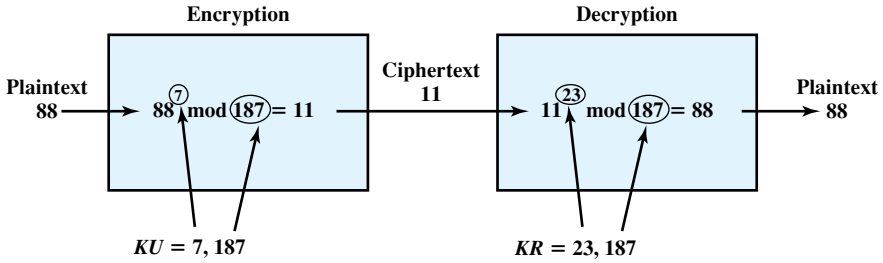

Figure 21.11 Example of RSA Algorithm

5. Determine d such that $de \bmod 160 = 1$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = 10 \times 160 + 1$.

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows:

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$
$$88^1 \bmod 187 = 88$$
$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$
$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$
$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$
$$11^1 \bmod 187 = 11$$
$$11^2 \bmod 187 = 121$$
$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$
$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$
$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

There are two possible approaches to defeating the RSA algorithm. The first is the brute-force approach: Try all possible private keys. Thus, the larger the number of bits in e and d, the more secure the algorithm. However, because the calculations involved, both in key generation and in encryption/decryption, are complex, the larger the size of the key, the slower the system will run.

Most discussions of the cryptanalysis of RSA have focused on the task of factoring n into its two prime factors. For a large n with large prime factors, factoring is a hard problem, but not as hard as it used to be. A striking illustration of this is the following. In 1977, the three inventors of RSA dared *Scientific American* readers to decode a cipher they printed in Martin Gardner's "Mathematical Games" column.

They offered a $100 reward for the return of a plaintext sentence, an event they predicted might not occur for some 40 quadrillion years. In April of 1994, a group working over the Internet and using over 1600 computers claimed the prize after only eight months of work [LEUT94]. This challenge used a public-key size (length of n) of 129 decimal digits, or around 428 bits. This result does not invalidate the use of RSA; it simply means that larger key sizes must be used. Currently, a 1024-bit key size (about 300 decimal digits) is considered strong enough for virtually all applications.

Key Management

With symmetric encryption, a fundamental requirement for two parties to communicate securely is that they share a secret key. Suppose Bob wants to create a messaging application that will enable him to exchange e-mail securely with anyone who has access to the Internet or to some other network that the two of them share. Suppose Bob wants to do this using only symmetric encryption. With symmetric encryption, Bob and his correspondent, say, Alice, must come up with a way to share a unique secret key that no one else knows. How are they going to do that? If Alice is in the next room from Bob, Bob could generate a key and write it down on a piece of paper or store it on a diskette and hand it to Alice. But if Alice is on the other side of the continent or the world, what can Bob do? Well, he could encrypt this key using symmetric encryption and e-mail it to Alice, but this means that Bob and Alice must share a secret key to encrypt this new secret key. Furthermore, Bob and everyone else who uses this new e-mail package faces the same problem with every potential correspondent: Each pair of correspondents must share a unique secret key.

How to distribute secret keys securely is the most difficult problem for symmetric encryption. This problem is wiped away with public-key encryption by the simple fact that the private key is never distributed. If Bob wants to correspond with Alice and other people, he generates a single pair of keys, one private and one public. He keeps the private key secure and broadcasts the public key to all and sundry. If Alice does the same, then Bob has Alice's public key, Alice has Bob's public key, and they can now communicate securely. When Bob wishes to communicate with Alice, Bob can do the following:

1. Prepare a message.
2. Encrypt that message using symmetric encryption with a one-time symmetric session key.
3. Encrypt the session key using public-key encryption with Alice's public key.
4. Attach the encrypted session key to the message and send it to Alice.

Only Alice is capable of decrypting the session key and therefore of recovering the original message.

It is only fair to point out, however, that we have replaced one problem with another. Alice's private key is secure because she need never reveal it; however, Bob must be sure that the public key with Alice's name written all over it is in fact Alice's public key. Someone else could have broadcast a public key and said it was Alice's.

The solution to this problem is the **public-key certificate**. In essence, a certificate consists of a public key plus a User ID of the key owner, with the whole

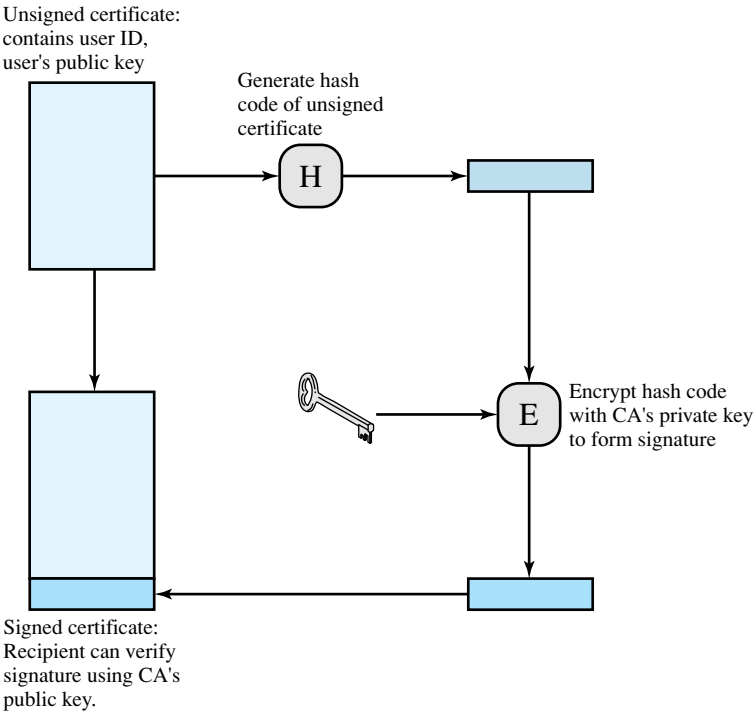

Figure 21.12 Public-Key Certificate Use

block signed by a trusted third party. Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. Figure 21.12 illustrates the process.

21.5 SECURE SOCKET LAYER AND TRANSPORT LAYER SECURITY

One of the most widely used security services is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS), the latter defined in RFC 2246. SSL is a general-purpose service implemented as a set of protocols that rely on TCP. At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

This section discusses SSLv3. Only minor changes are found in TLS.

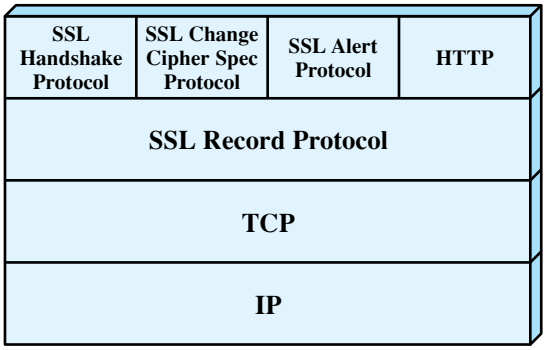

Figure 21.13 SSL Protocol Stack

SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure 21.13.

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, the Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL exchanges and are examined later in this section.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows:

- **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for symmetric encryption of SSL payloads.
- **Message integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

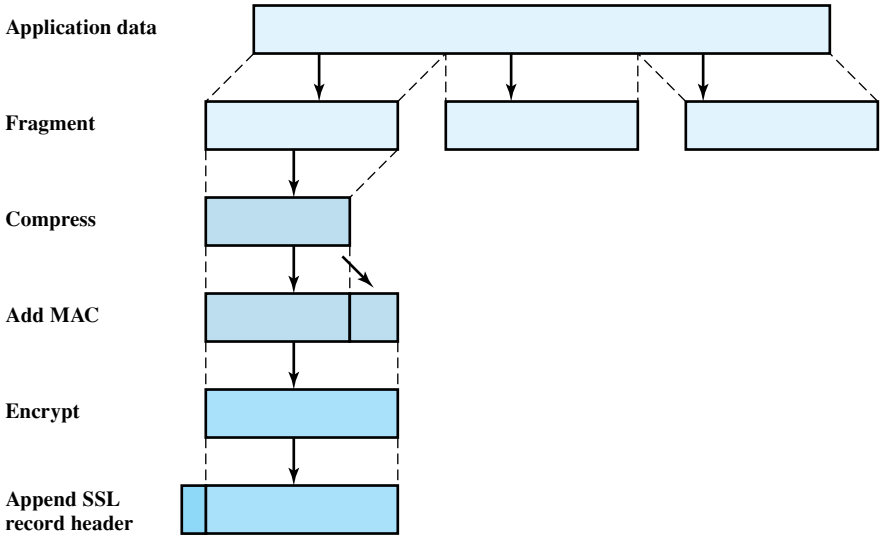

Figure 21.14 SSL Record Protocol Operation

Figure 21.14 indicates the overall operation of the SSL Record Protocol. The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16,384 bytes) or less. Next, **compression** is optionally applied. The next step in processing is to compute a **message authentication code** over the compressed data. Next, the compressed message plus the MAC are **encrypted** using symmetric encryption.

The final step of SSL Record Protocol processing is to prepend a header, consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is $2^{14} + 2048$.

The content types that have been defined are change_cipher_spec, alert, handshake, and application_data. The first three are the SSL-specific protocols, discussed next. Note that no distinction is made among the various applications (e.g., HTTP) that might use SSL; the content of the data created by such applications is opaque to SSL.

The Record Protocol then transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled and then delivered to higher-level users.

Change Cipher Spec Protocol

The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest. This protocol consists of a single message, which consists of a single byte with the value 1. The sole purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.

Each message in this protocol consists of two bytes. The first byte takes the value warning(1) or fatal(2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert. An example of a fatal alert is an incorrect MAC. An example of a nonfatal alert is a close_notify message, which notifies the recipient that the sender will not send any more messages on this connection.

Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted.

The Handshake Protocol consists of a series of messages exchanged by client and server. Figure 21.15 shows the initial exchange needed to establish a logical connection between client and server. The exchange can be viewed as having four phases.

Phase 1 is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a client_hello message with the following parameters:

- **Version:** The highest SSL version understood by the client.
- **Random:** A client-generated random structure, consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values are used during key exchange to prevent replay attacks.
- **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.
- **CipherSuite:** This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec.

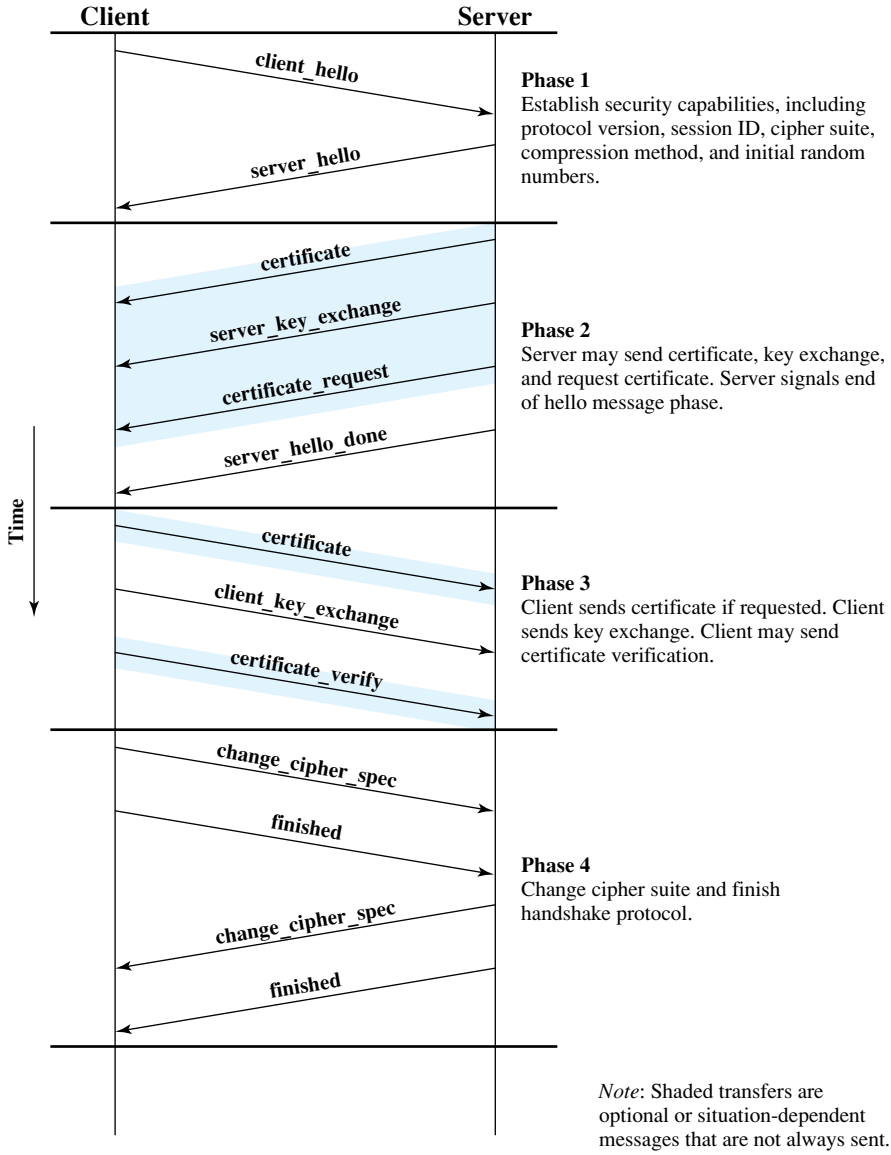

Figure 21.15 Handshake Protocol Action

- **Compression Method:** This is a list of the compression methods the client supports.

After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message.

The details of **phase 2** depend on the underlying public-key encryption scheme that is used. In some cases, the server passes a certificate to the client, possibly additional key information, and a request for a certificate from the client.

The final message in phase 2, and one that is always required, is the server_done message, which is sent by the server to indicate the end of the server hello and associated messages. After sending this message, the server will wait for a client response.

In **phase 3**, upon receipt of the server_done message, the client should verify that the server provided a valid certificate if required and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messages back to the server, depending on the underlying public-key scheme.

Phase 4 completes the setting up of a secure connection. The client sends a change_cipher_spec message and copies the pending CipherSpec into the current CipherSpec. Note that this message is not considered part of the Handshake Protocol but is sent using the Change Cipher Spec Protocol. The client then immediately sends the finished message under the new algorithms, keys, and secrets. The finished message verifies that the key exchange and authentication processes were successful.

In response to these two messages, the server sends its own change_cipher_spec message, transfers the pending to the current CipherSpec, and sends its finished message. At this point the handshake is complete and the client and server may begin to exchange application layer data.

21.6 IPv4 AND IPv6 SECURITY

In 1994, the Internet Architecture Board (IAB) issued a report entitled *Security in the Internet Architecture* (RFC 1636). The report stated the general consensus that the Internet needs more and better security, and it identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms.

These concerns are fully justified. The Computer Emergency Response Team (CERT) Coordination Center (CERT/CC) reports an ever-increasing number of Internet-related vulnerabilities (**http://www.cert.org**). These include security weaknesses in the operating systems of attached computers (e.g., Windows, Linux) as well as vulnerabilities in Internet routers and other network devices. Similarly CERT/CC documents a growing number of security-related incidents. These include denial of service attacks; IP spoofing, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP; and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents.

In response to these issues, the IAB included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with IPv4 and IPv6. This means that vendors can begin offering these features now, and many vendors do now have some IPSec capability in their products.

Applications of IPSec

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners:** IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

The principal feature of IPSec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

The Scope of IPSec

IPSec provides three main facilities: an authentication-only function referred to as Authentication Header (AH), a combined authentication/encryption function called Encapsulating Security Payload (ESP), and a key exchange function. For virtual private networks, both authentication and encryption are generally desired, because it is important both to (1) assure that unauthorized users do not penetrate the virtual private network and (2) assure that eavesdroppers on the Internet cannot read messages sent over the virtual private network. Because both features are generally desirable, most implementations are likely to use ESP rather than AH. The key exchange function allows for manual exchange of keys as well as an automated scheme.

The IPSec specification is quite complex and covers numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408. In this section, we provide an overview of some of the most important elements of IPSec.

Security Associations

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both.

A security association is uniquely identified by three parameters:

- **Security parameters index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

- **IP destination address:** Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end-user system or a network system such as a firewall or router.
- **Security protocol identifier:** This indicates whether the association is an AH or ESP security association.

Hence, in any IP packet, the security association is uniquely identified by the Destination Address in the IPv4 or IPv6 header and the SPI in the enclosed extension header (AH or ESP).

An IPSec implementation includes a security association database that defines the parameters associated with each SA. A security association is defined by the following parameters:

- **Sequence number counter:** A 32-bit value used to generate the sequence number field in AH or ESP headers.
- **Sequence counter overflow:** A flag indicating whether overflow of the sequence number counter should generate an auditable event and prevent further transmission of packets on this SA.
- **Antireplay window:** Used to determine whether an inbound AH or ESP packet is a replay, by defining a sliding window within which the sequence number must fall.
- **AH information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH.
- **ESP information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP.
- **Lifetime of this security association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur.
- **IPSec protocol mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section.
- **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

The key management mechanism that is used to distribute keys is coupled to the authentication and privacy mechanisms only by way of the security parameters index. Hence, authentication and privacy have been specified independent of any specific key management mechanism.

Authentication Header

The authentication header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet.

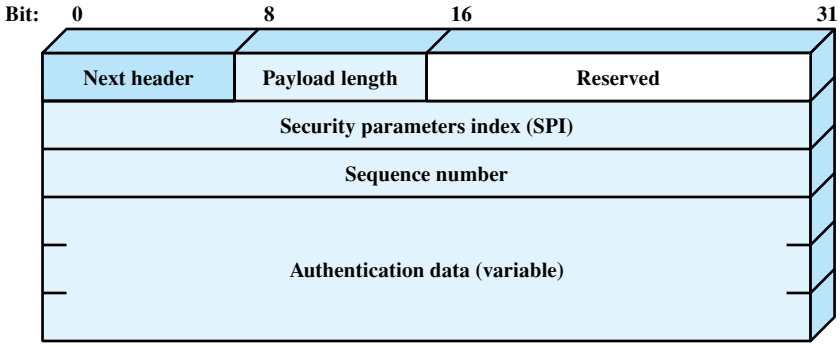

Figure 21.16 IPSec Authentication Header

Authentication is based on the use of a message authentication code (MAC), as described in Section 21.3; hence the two parties must share a secret key.

The authentication header consists of the following fields (Figure 21.16):

- **Next Header (8 bits):** Identifies the type of header immediately following this header.
- **Payload Length (8 bits):** Length of authentication header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4.
- **Reserved (16 bits):** For future use.
- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value.
- **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the integrity check value (ICV), or MAC, for this packet.

The authentication data field is calculated over

- IP header fields that either do not change in transit (immutable) or that are predictable in value upon arrival at the endpoint for the AH SA. Fields that may change in transit and whose value on arrival are unpredictable are set to zero for purposes of calculation at both source and destination.
- The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source and destination.
- The entire upper-level protocol data, which is assumed to be immutable in transit.

For IPv4, examples of immutable fields are Internet Header Length and Source Address. An example of a mutable but predictable field is the Destination Address (with loose or strict source routing). Examples of mutable fields that are zeroed prior to ICV calculation are the Time to Live and Header Checksum fields. Note that both source and destination address fields are protected, so that address spoofing is prevented.

For IPv6, examples in the base header are Version (immutable), Destination Address (mutable but predictable), and Flow Label (mutable and zeroed for calculation).

Encapsulating Security Payload

The encapsulating security payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

Figure 21.17 shows the format of an ESP packet. It contains the following fields:

- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value.
- **Payload Data (variable):** This is an upper-level segment protected by encryption.
- **Padding (0–255 bytes):** May be required if the encryption algorithm requires the plaintext to be a multiple of some number of octets.
- **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
- **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).
- **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the integrity check value computed over the ESP packet minus the Authentication Data field.

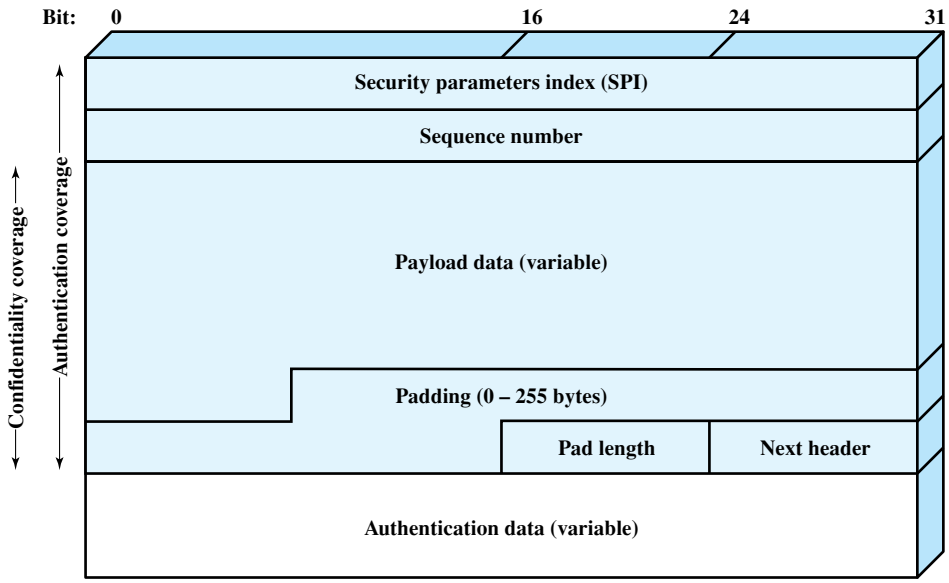

Figure 21.17 IPSec ESP Format

21.7 Wi-Fi PROTECTED ACCESS

As discussed in Section 17.6, the 802.11i task group has developed a set of capabilities to address the WLAN security issues. In order to accelerate the introduction of strong security into WLANs, the Wi-Fi Alliance promulgated **Wi-Fi Protected Access (WPA)** as a Wi-Fi standard. WPA is a set of security mechanisms that eliminates most 802.11 security issues and was based on the current state of the 802.11i standard. As 802.11i evolves, WPA will evolve to maintain compatibility.

IEEE 802.11i addresses three main security areas: authentication, key management, and data transfer privacy. To improve authentication, 802.11i requires the use of an authentication server (AS) and defines a more robust authentication protocol. The AS also plays a role in key distribution. For privacy, 802.11i provides three different encryption schemes. The scheme that provides a long-term solution makes use of the Advanced Encryption Standard (AES) with 128-bit keys. However, because the use of AES would require expensive upgrades to existing equipment, alternative schemes based on 104-bit RC4 are also defined.

Figure 21.18 gives a general overview of 802.11i operation. First, an exchange between a station and an AP enables the two to agree on a set of security capabilities to be used. Then, an exchange involving the AS and the station provides for secure authentication. The AS is responsible for key distribution to the AP, which in turn manages and distributes keys to stations. Finally, strong encryption is used to protect data transfer between the station and the AP.

The 802.11i architecture consists of three main ingredients:

- **Authentication:** A protocol is used to define an exchange between a user and an AS that provides mutual authentication and generates temporary keys to be used between the client and the AP over the wireless link.

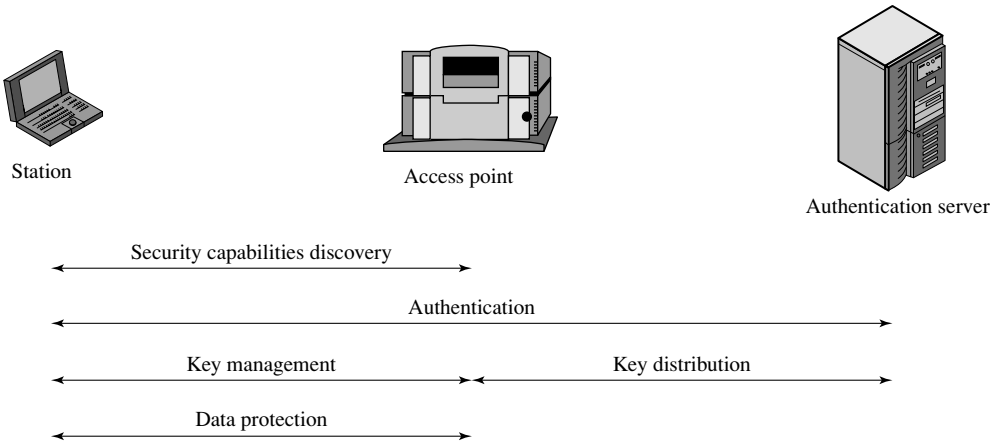

Figure 21.18 802.11i Operational Phases

- **Access control:** This function enforces the use of the authentication function, routes the messages properly, and facilitates key exchange. It can work with a variety of authentication protocols.
- **Privacy with message integrity:** MAC-level data (e.g., an LLC PDU) are encrypted, along with a message integrity code that ensures that the data have not been altered.

Authentication operates at a level above the LLC and MAC protocols and is considered beyond the scope of 802.11. There are a number of popular authentication protocols in use, including the Extensible Authentication Protocol (EAP) and the Remote Authentication Dial-In User Service (RADIUS). These are not covered in this book. The remainder of this section examines access control and privacy with message integrity.

Access Control[5]

IEEE 802.11i makes use of another standard that was designed to provide access control functions for LANs. The standard is IEEE 802.1X, Port-Based Network Access Control. IEEE 802.1X uses the terms *supplicant, authenticator*, and *authentication server* (AS). In the context of an 802.11 WLAN, the first two terms correspond to the wireless station and the AP. The AS is typically a separate device on the wired side of the network (i.e., accessible over the DS) but could also reside directly on the authenticator.

Before a supplicant is authenticated by the AS, using an authentication protocol, the authenticator only passes control or authentication messages between the supplicant and the AS; the 802.1X control channel is unblocked but the 802.11 data channel is blocked. Once a supplicant is authenticated and keys are provided, the authenticator can forward data from the supplicant, subject to predefined access control limitations for the supplicant to the network. Under these circumstances, the data channel is unblocked.

As indicated in Figure 21.19, 802.1X uses the concepts of controlled and uncontrolled ports. Ports are logical entities defined within the authenticator and refer to physical network connections. For a WLAN, the authenticator (the AP) may have only two physical ports, one connecting to the DS and one for wireless communication within its BSS. Each logical port is mapped to one of these two physical ports. An uncontrolled port allows the exchange of PDUs between the supplicant and other the AS regardless of the authentication state of the supplicant. A controlled port allows the exchange of PDUs between a supplicant and other systems on the LAN only if the current state of the supplicant authorizes such an exchange.

The 802.1X framework, with an upper-layer authentication protocol, fits nicely with a BSS architecture that includes a number of wireless stations and an AP. However, for an IBSS, there is no AP. For an IBSS, 802.11i provides a more complex solution that, in essence, involves pairwise authentication between stations on the IBSS.

[5]In this subsection, we are discussing access control as a security function. This is a different function than medium access control (MAC) as described in Chapter 15. Unfortunately, the literature and the standards use the term *access control* in both contexts.

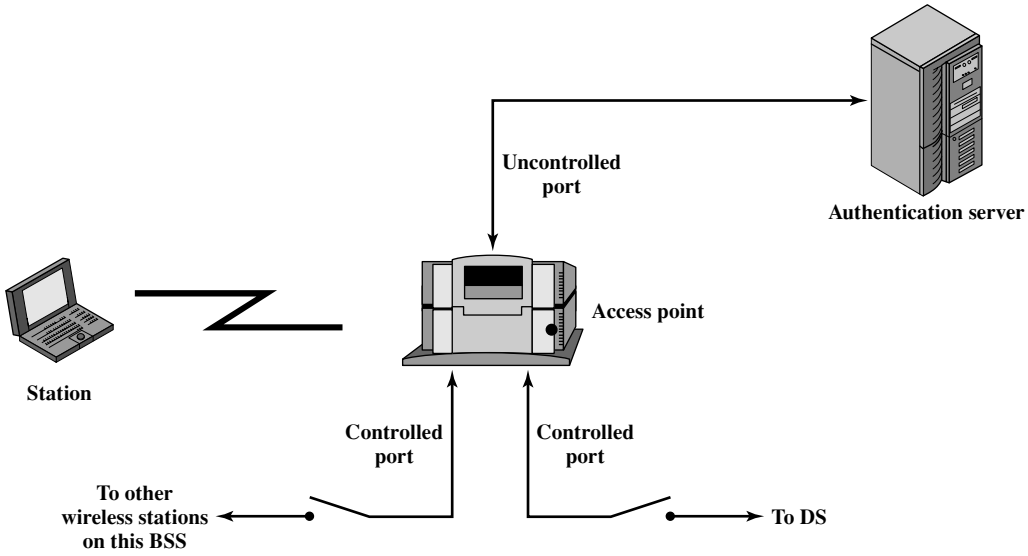

Figure 21.19 802.11i Access Control

Privacy with Message Integrity

IEEE 802.11i defines two schemes for protecting data transmitted in 802.11 MAC PDUs. The first scheme is known as the Temporal Key Integrity Protocol (TKIP) or WPA-1. TKIP is designed to require only software changes to devices that are implemented with an older wireless LAN security approach called Wired Equivalent Privacy (WEP); it uses the same RC4 stream encryption algorithm as WEP. The second scheme is known as Counter Mode-CBC MAC Protocol (CCMP) or WPA-2. CCMP makes use of the Advanced Encryption Standard (AES) encryption protocol.[6]

Both TKIP and WPA-2 add a message integrity code (MIC) to the 802.11 MAC frame after the data field. The MIC is generated by an algorithm, called Michael, that computes a 64-bit value calculated using the source and destination MAC address values and the Data field. This value is then encrypted using a separate key from that used for encrypting the Data fields. Thus, both the data and MIC fields are encrypted. The use of a more complex algorithm, a separate encryption key, and a 64-bit length, all make the MIC and substantially stronger message authentication feature than the ICV. The MIC serves the purpose of message authentication, a function described earlier in this chapter.

21.8 RECOMMENDED READING AND WEB SITES

The topics in this chapter are covered in greater detail in [STAL06]. For coverage of cryptographic algorithms, [SCHN96] is a valuable reference work; it contains descriptions of virtually every cryptographic algorithm and protocol in use up to the time of the book's publication.

[6]The AES algorithm is described in detail in [STAL06] and [STAL02].

SCHN96 Schneier, B. *Applied Cryptography.* New York: Wiley, 1996.

STAL06 Stallings, W. *Cryptography and Network Security: Principles and Practice, Fourth Edition.* Upper Saddle River, NJ: Prentice Hall, 2003.

Recommended Web sites:

- **Coast:** Comprehensive set of links related to cryptography and network security
- **IETF Security Area:** Provides up-to-date information on Internet security standardization efforts
- **IEEE Technical Committee on Security and Privacy:** Provides copies of IEEE's newsletter and information on IEEE-related activities

21.9 KEY TERMS, REVIEW QUESTIONS, AND PROBLEMS

Key Terms

active attack	hash function	public-key certificate
authenticity	integrity	public-key encryption
availability	IP Security (IPSec)	replay
Advanced Encryption Standard (AES)	key distribution	RSA
brute-force attack	key distribution center	secret key
ciphertext	key management	secure hash function
confidentiality	masquerade	Secure Socket Layer (SSL)
cryptanalysis	message authentication	session key
Data Encryption Standard (DES)	message authentication code (MAC)	SHA
decryption algorithm	one-way hash function	symmetric encryption
denial of service	passive attack	traffic analysis
digital signature	plaintext	traffic padding
encryption algorithm	private key	Transport Layer Security (TLS)
	public key	

Review Questions

21.1 What is the difference between passive and active security threats?
21.2 List and briefly define categories of passive and active security threats.
21.3 What are DES and triple DES?
21.4 How is the AES expected to be an improvement over triple DES?
21.5 Explain traffic padding.
21.6 List and briefly define various approaches to message authentication.
21.7 What is a secure hash function?

21.8 Explain the difference between symmetric encryption and public-key encryption.
21.9 What are the distinctions among the terms *public key, private key, secret key*?
21.10 What is a digital signature?
21.11 What is a public-key certificate?
21.12 What protocols comprise SSL?
21.13 What is the difference between and SSL connection and an SSL session?
21.14 What services are provided by the SSL Record Protocol?
21.15 What services are provided by IPSec?

Problems

21.1 Give some examples where traffic analysis could jeopardize security. Describe situations where end-to-end encryption combined with link encryption would still allow enough traffic analysis to be dangerous.

21.2 Key distribution schemes using an access control center and/or a key distribution center have central points vulnerable to attack. Discuss the security implications of such centralization.

21.3 Suppose that someone suggests the following way to confirm that the two of you are both in possession of the same secret key. You create a random bit string the length of the key, XOR it with the key, and send the result over the channel. Your partner XORs the incoming block with the key (which should be the same as your key) and sends it back. You check and if what you receive is your original random string, you have verified that your partner has the same secret key, yet neither of you has ever transmitted the key. Is there a flaw in this scheme?

21.4 Prior to the discovery of any specific public-key schemes, such as RSA, an existence proof was developed whose purpose was to demonstrate that public-key encryption is possible in theory. Consider the functions $f_1(x_1) = z_1; f_2(x_2, y_2) = z_2; f_3(x_3, y_3) = z_3$, where all values are integers with $1 \leq x_i, y_i, z_i \leq N$. Function f_1 can be represented by a vector M1 of length N, in which the kth entry is the value of $f_1(k)$. Similarly, f_2 and f_3 can be represented by $N \times N$ matrices M2 and M3. The intent is to represent the encryption/decryption process by table lookups for tables with very large values of N. Such tables would be impractically huge but could, in principle, be constructed. The scheme works as follows: Construct M1 with a random permutation of all integers between 1 and N; that is, each integer appears exactly once in M1. Construct M2 so that each row contains a random permutation of the first N integers. Finally, fill in M3 to satisfy the following condition:

$$f_3(f_2(f_1(k), p), k) = p \text{ for all } k, p \text{ with } 1 \leq k, p \leq N$$

In words,

1. M1 takes an input k and produces an output x.
2. M2 takes inputs x and p giving output z.
3. M3 takes inputs z and k and produces p.

The three tables, once constructed, are made public.

a. It should be clear that it is possible to construct M3 to satisfy the preceding condition. As an example, fill in M3 for the following simple case:

M1 =

5
4
2
3
1

M2 =

5	2	3	4	1
4	2	5	1	3
1	3	2	4	5
3	1	4	2	5
2	5	3	4	1

M3 =

Convention: The ith element of M1 corresponds to $k = i$. The ith row of M2 corresponds to $x = i$; the jth column of M2 corresponds to $p = j$. The ith row of M3 corresponds to $z = i$; the jth column of M3 corresponds to $k = j$.

b. Describe the use of this set of tables to perform encryption and decryption between two users.

c. Argue that this is a secure scheme.

21.5 Perform encryption and decryption using the RSA algorithm, as in Figure 21.11, for the following:

a. $p = 3; q = 11, d = 7; M = 5$

b. $p = 5; q = 11, e = 3; M = 9$

c. $p = 7; q = 11, e = 17; M = 8$

d. $p = 11; q = 13, e = 11; M = 7$

e. $p = 17; q = 31, e = 7; M = 2$. *Hint*: Decryption is not as hard as you think; use some finesse.

21.6 In a public-key system using RSA, you intercept the ciphertext $C = 10$ sent to a user whose public key is $e = 5, n = 35$. What is the plaintext M?

21.7 In an RSA system, the public key of a given user is $e = 31, n = 3599$. What is the private key of this user?

21.8 Suppose we have a set of blocks encoded with the RSA algorithm and we don't have the private key. Assume $n = pq$, e is the public key. Suppose also someone tells us he or she knows one of the plaintext blocks has a common factor with n. Does this help us in any way?

21.9 Show how RSA can be represented by matrices M1, M2, and M3 of Problem 21.4.

21.10 Consider the following scheme:

1. Pick an odd number, E.
2. Pick two prime numbers, P and Q, where $(P - 1)(Q - 1) - 1$ is evenly divisible by E.
3. Multiply P and Q to get N.
4. Calculate $D = \dfrac{(P - 1)(Q - 1)(E - 1) + 1}{E}$.

Is this scheme equivalent to RSA? Show why or why not.

21.11 Consider using RSA with a known key to construct a one-way hash function. Then process a message consisting of a sequence of blocks as follows: Encrypt the first block, XOR the result with the second block and encrypt again, and so on. Show that this scheme is not secure by solving the following problem. Given a two-block message B1, B2, and its hash

$$\text{RSAH}(\text{B1}, \text{B2}) = \text{RSA}(\text{RSA}(\text{B1}) \oplus \text{B2})$$

Given an arbitrary block C1, choose C2 so that $\text{RSAH}(\text{C1}, \text{C2}) = \text{RSAH}(\text{B1}, \text{B2})$.

21.12 In SSL and TLS, why is there a separate Change Cipher Spec Protocol rather than including a change_cipher_spec message in the Handshake Protocol?

21.13 In discussing AH processing, it was mentioned that not all of the fields in an IP header are included in MAC calculation.

a. For each of the fields in the IPv4 header, indicate whether the field is immutable, mutable but predictable, or mutable (zeroed prior to ICV calculation).

b. Do the same for the IPv6 header.

c. Do the same for the IPv6 extension headers.

In each case, justify your decision for each field.