CHAPTER 5

IP SECURITY

MOSTAFA HASHEM SHERIF

5.1 INTRODUCTION

The goal of this chapter is to review the general principles of security in IP networks as a particular case of telecommunications networks. This chapter does not give an exhaustive treatment of the subject. It gives a description of security services in open networks and describes various security mechanisms using cryptography. Architectures for certification as well as management of encryption keys are presented. Some potential threats to security are highlighted, particularly as they relate to cracks in the protection walls of cryptography.

The chapter has four appendices. The first lists some policy considerations for securing telecommunication networks. A general overview of the symmetric and public key encryption algorithms is available in Appendices II and III, respectively. Appendix IV describes the main operations of the Digital Signature Algorithm (DSA) of ANSI X9.30:1 [1].

5.2 SECURITY OF TELECOMMUNICATIONS SERVICES

Telecommunications services build on networking technologies, operations support systems, and policies (also called methods and procedures) to allow remote access to applications as well as content management and distribution. Within this context, Recommendations X.700 [2] and X.800 [3] of the ITU-T provide a general framework to ensure the security of telecommunications services. First, security functions or services must be available. Next, these security functions must be securely managed. Finally, policies

Managing IP Networks. Edited by Saleh Aidarous and Thomas Plevyak
ISBN 0-471-39299-5

should be defined and implemented to administer the security information and the information base, as well as to ensure the physical security of infrastructure (protection from fires, earthquakes, intrusions, attacks, thefts, etc.).

The complexity of securing telecommunications services today arises from several factors. Because of the worldwide phenomenon of deregulation of telecommunications, service offers are no longer vertically integrated and several operators usually collaborate to carry the traffic end-to-end. As shown in Figure 5.1, current offers of telecommunications services involve several providers end-to-end for the following components: the infrastructure for transport, the switching functions within the network, the network services; and the content management, including access, distribution, billing, and payment collection. For example, service providers can be involved in Web hosting, in disaster recovery, or in data storage networks. Content management covers the functions of customer relations management, supply management, and electronic payments. The content-level providers include broadcasters, providers of catalogues, and certification authorities. In all these instances, when a problem arises, operators at each layer need to be able to exchange authenticated information that would help in locating and resolving this problem [4, 5]. Furthermore, VPNs may span several carriers which should collaborate to ensure the continuity of service from one end to the other, even though each carrier manages its own part separately. In VPNs, customer network management (CNM) systems, such as those defined by ITU-T Recommendations X.160, X.161, and X.162 [6–8], allow end users to have controlled access to their part of the public data network for monitoring and provisioning. Clearly, the correct identification and authentication of the participants, their administrative privileges, and their traffic are essential to provide them with the services to which they have subscribed, to ensure the integrity and confidentiality of the exchanges, and to prevent users from affecting other users (inadvertently or out of malice).

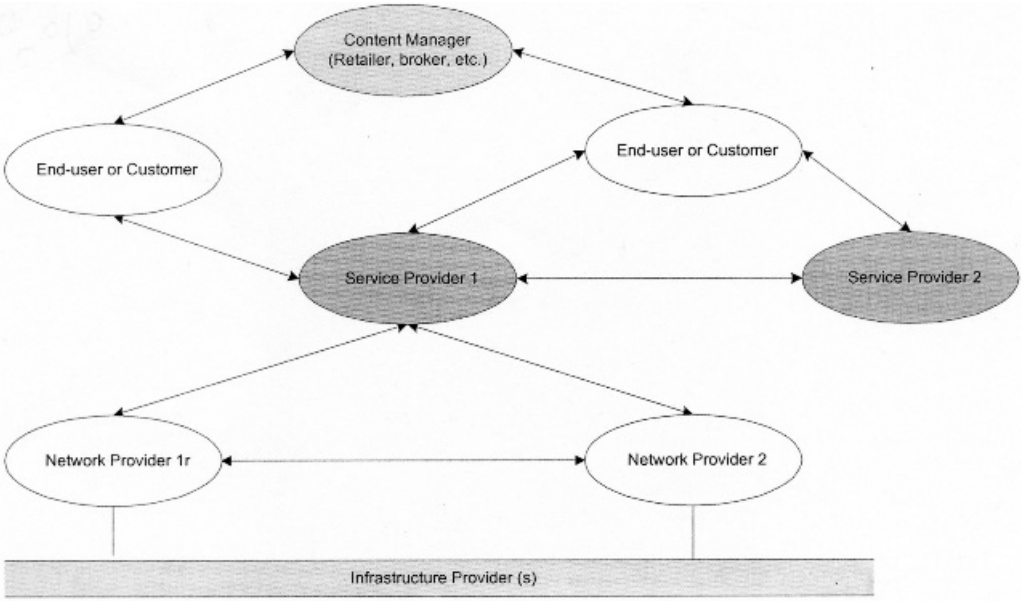

Figure 5.1 Providers of telecommunications services.

Within each network, access management to the Network Operations Center (NOC) is complicated by the fact that hundreds or technicians and applications perform specific operations. Although automation of operations is pursued to reduce cost, it requires that various OSSs communicate securely, even across administrative boundaries. It will also be necessary to preserve records of the exchanges as proof that can help resolve disputes and litigation [9]. Clearly, security in such a distributed and complex environment depends, not only on network equipment (switches, transmit trunks, and information systems), but also on the end user's terminals and the administrative policies of each operator. This is why the management systems must at least incorporate highly reliable security mechanisms to protect the telecommunication services offers.

One key assumption in this chapter is that the communications network is continuously available. Attacks on the network infrastructure, for example, the signaling, routing, or network management mechanisms, increase security exposures. Therefore, the network infrastructure must be physically protected from fires, earthquakes, floods, vandalism, etc. It also means that the network elements have been thoroughly tested to ensure the correct routing of messages and the correct functioning of various network management functions, including operations and administration, under a wide range of loads and stress conditions. ISO has issued a technical report ISO/IEC TR 13335 [10] to provide guidance on the management aspects of information security. Part 5 of this report deals with communication networks and the factors that should be taken into account to establish network security requirements. Aspects related to physical protection, software quality evaluation, risk analysis, as well as recommended security policies are outside the scope of this chapter.

5.3 SECURITY OBJECTIVES

Security exposures can affect user data and applications, the network infrastucture of the network elements themselves. Recommendations X.509 [11] and X.800 [3] of the ITU-T identify several types of information threats that can be classified as follows.

1. Passive attacks
   —Interception of the identity of one or more of the participants by a third party with a mischievous intent;
   —Data interception through clandestine monitoring of the exchanges during a communication by an outsider or an unauthorized user.
2. Active attacks
   —Replay of a previous message, in its entirety or in part, after its recording;
   —Defective or criminal manipulation of the content of an exchange by substitution, insertion, deletion, reorganization of user's data exchanged in a communication by a nonauthorized third party;
   —Users' repudiation or denial of their participation in part or in all of a communication exchange;
   —Misrouting of messages from one user to another (the objective of the security service would be to avoid the consequences of such an error);
   —Analysis of the traffic and examination of the parameters related to a communication among users (i.e., absence or presence, frequency, direction, sequence,

type, volume, etc.). This analysis would be made more difficult with the production of unintelligible additional traffic (by a fill-in traffic) and by using encrypted or random data.

—Masquerade, whereby one entity pretends to be another entity;

—Denial of service and the impossibility of accessing the resources usually available to authorized users following the prevention or interruption of a communication, or the delay imposed on time-critical operations.

Based on the preceding threats, the objectives of security measures are as follows.

- Prevent an outsider other than the participants from reading or manipulating the contents or the sequences of the exchanged messages without being detected. In particular, this third party must not be allowed to play back old messages, replace blocks of information, or insert messages from multiple distinct exchanges without detection.
- Impede the falsification of payment instructions or the generation of spurious messages by users with dubious intentions. For example, dishonest merchants or processing centers must not be capable of reutilizing information about their clients' bank accounts to generate fraudulent orders. They should not be able to initiate the processing of payment instructions without expediting the corresponding purchases. At the same time, the merchants will be protected from excessive revocation of payments or malicious denials of orders.
- Satisfy the legal requirements for valid contracts to allow conflict resolutions, particularly in the area of consumer protection and privacy protection.
- Assure access to the service according to the contractual terms.
- Give the same level of service to all customers, irrespective of their location and the variations in climate, temperature, humidity, erosion, etc.

5.4 OSI MODEL FOR CRYPTOGRAPHIC SECURITY

The well-known OSI reference model of data networks establishes a structure for exchanges in seven layers, as follows:

1. The *physical layer,* where the electrical, mechanical, and functional properties of the interfaces are defined (signal levels, rates, structures, etc.).
2. The *link layer,* which defines the methods for orderly and error-free transmission between two network nodes.
3. The *network layer,* where the functions for routing, multiplexing of packets, flow control, and network supervision are defined.
4. The *transport layer,* which is responsible for the reliable transport of the traffic between the two network endpoints as well the assembly and disassembly of the messages.
5. The *session layer,* which handles the conversation between the processes at the two endpoints.

6. The *presentation layer,* which is in charge of managing the differences in syntax among the various representations of information at both endpoints by putting the data into a standardized format.
7. The *application layer,* whose function is to ensure that two application processes cooperate to carry out the desired information processing at the two endpoints.

The ISO standard ISO 7498 Part 2 [12] (ITU-T Recommendation X.800 [3]) describes a reference model for security services in open networks. Each layer of the ISO model can offer one or more of the following security services [13]:

- *Confidentiality,* so that the exchanged messages are not divulged to an nonauthorized third party. In some applications, the confidentiality of addresses may be needed as well, to prevent the analysis of traffic patterns and the derivation of side information that could be used.
- *Integrity* of the data, i.e., proof that the message has not been altered after it was expedited and before the moment it was received. This service guarantees that the received data are exactly what have been transmitted by the sender and that they have not been corrupted, either intentionally or by error in transit in the network. Data integrity is also needed for network management data such as configuration files, and accounting and audit information.
- *Identification* of the participants by verifying a preestablished relation between a characteristic (for example, a password or cryptographic key) and an entity. This allows control of access to the network resources or to the offered services based on the privileges associated with a given identity. One entity may possess several distinct identifiers. Furthermore, some protection against denial of service attacks can be achieved using access control.
- *Authentication* of the participants (users, network elements, and network element systems), which is the corroboration of the identity that an entity claims with the guarantee of a trusted third party. Authentication is necessary to assure nonrepudiation of users as well of network elements.
- *Access control* to ensure that only the authorized participants whose identities have been duly authenticated can gain access to the protected resources.
- *Nonrepudiation,* which is the service that offers proof that the integrity of the data and of their origin in an irrefutable relation that can be verified by a third party, for example, the nonrepudiation that the sender has sent the message or that a receiver has received the message. This service can be also called authentication of the origin of the data.

Unfortunately, not all the services offered on the Internet can be easily protected. The case of mobile IP illustrates this point. According to this protocol, a mobile node outside the zone that its home agent serves must register with the foreign agent in whose region it is currently located. Yet the protocol does not provide the means to authenticate the foreign agent by initiating the exchange of the secret key that will be used to protect the resubscription data [14, pp. 134–139, 189–192].

Security services can be implemented in one or more layers of the OSI model [15–17]. The choice of the layer depends on the following criteria:

1. If the protection has to be accorded to all the traffic flow in a uniform manner, the intervention has to be at the physical or the link layers. The only cryptographic service that is available at this level is confidentiality by encrypting the data or similar means (frequency hopping, spread spectrum, etc.). The protection of the traffic at the physical layer covers all the flow, not only user data but also the information related to network administration: alarms, synchronization, update of routing table, etc. The disadvantage of the protection at this level is that a successful attack will destabilize the whole security structure, because the same key is utilized for all transmissions. At the link layer, encryption can be end-to-end, based on the source/destination, provided that the same technology is used all the way through.
2. For a selective bulk protection that covers all the communications associated with a particular subnetwork from one end-system to another end-system, network layer encipherment will be chosen. Security at the network layer is also needed to secure the communication among the network elements, particularly for link-state protocols, such as OSPF or PNNI, where updates to the routing tables are automatically generated based on received information that is then flooded to the rest of the network.
3. For a protection with recovery after a fault, or if the network is not reliable, the security services will be at the transport layer. The services of this layer apply end-to-end either singly or in combination. These services are authentication—whether *simple* by passwords or *strong* by signature mechanisms or certificates—access control, confidentiality, and integrity.
4. If a high granularity of protection is required or if the nonrepudiation service has to be assured, the encryption will be at the application layer. It is at this level that most of the security protocols for commercial systems operate, which frees them from a dependency on the lower layers. All security services are available.

It should be noted that there are no services at the session layer. In contrast, the services offered at the presentation layer are confidentiality, which can be selective such as by a given data field, authentication, integrity (in whole or in part), and nonrepudiation with a proof of origin or proof of delivery.

The secure sockets layer (SSL)/transport layer security (TLS) protocols are widely used to secure the connection between a client and a server [18, 19]. With respect to the OSI reference model, SSL/TLS lie between the transport layer and the application layer.

Nevertheless, it may be sufficient for an attacker to discover that a communication is taking place among partners, and then attempt to guess, for example:

- The characteristics of the goods or services exchanged;.
- The conditions for acquisition: delivery intervals, conditions, and means of settlement;
- The financial settlement.

The establishment of an enciphered channel or "tunnel" between two points at the network layer can constitute a shield against such types of attack. It should be noticed, however, that other clues, such as the relative time to execute the cryptographic operations, or the variations in the electric consumption or the electromagnetic radiation, can permit an

analysis of the encrypted traffic and ultimately lead to breaking of the encryption algorithms [20].

5.4.1 Security Services at the Link Layer

RFC 1661 [21] defines the link-layer protocol PPP to carry traffic between two entities identified with their respective IP addresses. The Layer 2 Tunneling Protocol (L2TP), defined in RFC 2661 [22], extends the PPP operation by separating the processing of IP packets within the PPP frames from that of the traffic flowing between the two ends at the link layer. This distinction allows a remote client to connect to a network access server (NAS) in a private (corporate) network though the public Internet as follows. The client encapsulates PPP frames in an L2TP tunnel, prepends the appropriate L2TP header, and then transports the new IP packet using UDP. The IP addresses in the new IP header are assigned by the local ISP at the local access point. Figure 5.2 illustrates the arrangement where the size of the additional header ranges from 8 octets to 16 octets: 1–2 octets for PPP, 8–16 octets for L2TP. Given that the overhead for UDP is 8 octets and for the IP header it is 20 octets, the total additional overhead ranges from 37 octets to 46 octets.

Although L2TP does not provide any security services, it is possible to use IPSec to secure the layer 2 tunnel, because L2TP runs over IP. This is shown in the following section.

5.4.2 Security Services at the Network Layer

The security services at this layer are offered from one end of the network to the other. They include network access control, authentication of the users and/or hosts, and authentication and integrity of the exchanges. These services are transparent to applications and end users, and their responsibilities fall on the administrators of network elements.

The purpose of network access control is to limit actions and privileges of an entity based on network addresses of both endpoints (e.g., IP addresses). As explained earlier, this is important in link-state protocols, such as OSPF or PNNI, to protect routing tables of various network elements.

Authentication at the network layer can be simple or strong. *Simple* authentication uses a name and password pair (the password can be a one-time password), while *strong* authentication utilizes digital signatures or the exchange of certificates issued by a recognized certification authority. The use of strong authentication requires the presence of encryption keys at all network nodes, which imposes the physical protection of all these nodes.

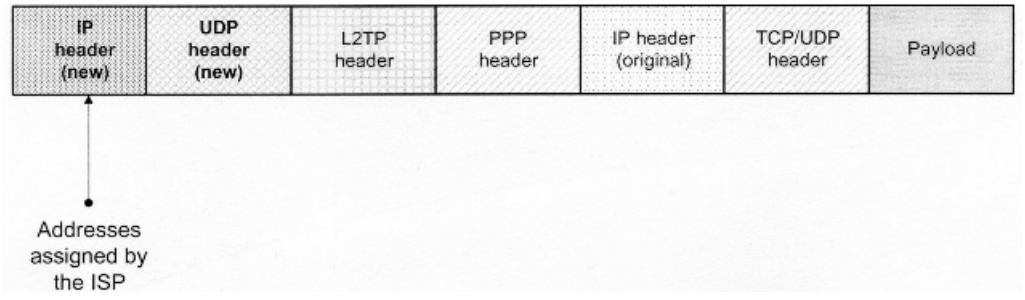

Figure 5.2 Layer 2 Tunneling with L2TP.

IPSEC is a protocol suite defined in RFCs 2401 to 2412 [23–31] to secure communications at the network layer between two peers. The overall security architecture is described in RFC 2401 [23], while a road map to the IPSec documentation is in RFC 2411 [29].

IPSec offers authentication, confidentiality, and key management. The authentication header (AH) protocol defined in RFC 2402 [23] provides the cryptographic services to authenticate and verify the integrity of the payload as well as the routing information in the original IP header. The encapsulating security payload (ESP) protocol is described in RFC 2406 [24], and gives the means to assure the confidentiality of the original payload and to authenticate the encrypted data as well as the ESP header. Both IPSec protocols provide some protection against replay attacks with the help of a monotonically increasing sequence number that is 32 bits long. Although these two mechanisms are available, in the IP version 6 (IPv6) protocol [32], IPSec makes them available with the current IP version 4. The key exchange is performed with the Internet key exchange (IKE) protocol defined in RFC 2409 [28]. (*Note:* A new ESP draft uses 64-bit sequence numbers and takes into consideration the new symmetric encryption algorithm Advance Encryption Standard (AES).)

IPSec operates in one of two modes: the transport mode and the tunnel mode. In the transport mode, protection covers the payload and transport header only, while the tunnel mode protects the whole packet, including IP addresses. The transport mode secures communication between two hosts, while the tunnel mode is useful when one or both ends of the connection is a trusted entity, such as a firewall, which provides security services to an originating device. Tunnel mode is also employed when a router provides security services to the traffic that it is forwarding [33]. Both modes are used to secure virtual private networks with IPSec, as shown in Figure 5.3. Typically, AH protocol can be used for the transport mode, while the ESP is applicable to both modes. This explains why there is a decreasing tendency to use the AH protocol.

Figure 5.4 illustrates the encapsulation in both cases. In this figure, the IPSec header represents either the ESP or both the ESP and the AH headers. Thus, routing information associated with the private or corporate network can be encrypted after establishment of a TCP tunnel between the firewall at the originating side and the one at the destination side. (*Note:* ESP with no encryption (i.e., with a NULL algorithm) is equivalent to the AH protocol, which is another reason why usage of the AH protocol is limited.)

In verifying the integrity, the contents of fields in the IP header that change in transit (e.g., the “time to live”) are considered to be zero. With respect to transmission overheads, the length of the AH is at least 12 octets (a multiple of 4 octets for IPv4 and of 6 octets for IPv6). Similarly, the length of the ESP header is 8 octets. However, the overhead includes 4 octets for the initialization vector (if it is included in the payload field) as well as an ESP trailer of at least 6 octets that comprise a padding and authentication data.

Let us return back to the protection of L2TP (control data or user information) traffic with the IPSec protocol suite, as described in RFC 3193 [34]. When both IPSec and L2TP are used together, the various headers are organized as shown in Figure 5.5. (*Note:* In the 1996–1998 time frame, RSA Data Security, Inc. (RSADSI), and the Secure Wide Area Network (S/WAN) consortium were actively promoting a specific implementation of IPSec to ensure interoperability among firewalls and TCP/IP products. However, the free-software advocates cooperated under the umbrella of FreeS/WAN to distribute an open-source implementation of both IPSec and its default exchange protocol IKE written for

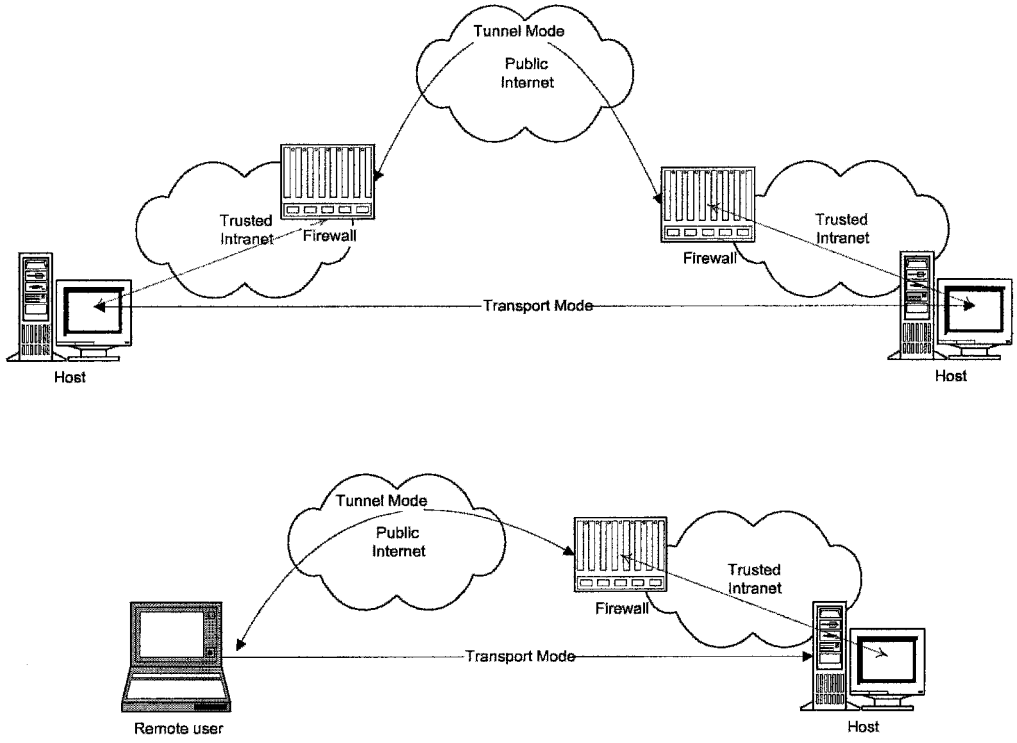

Figure 5.3 Securing virtual private networks with IPSec.

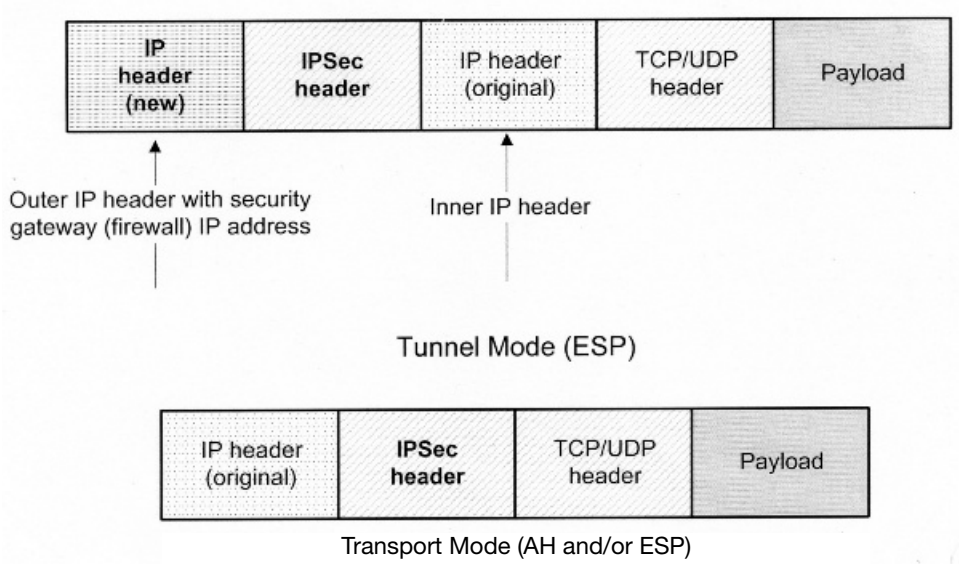

Figure 5.4 Encapsulation for IPSec modes.

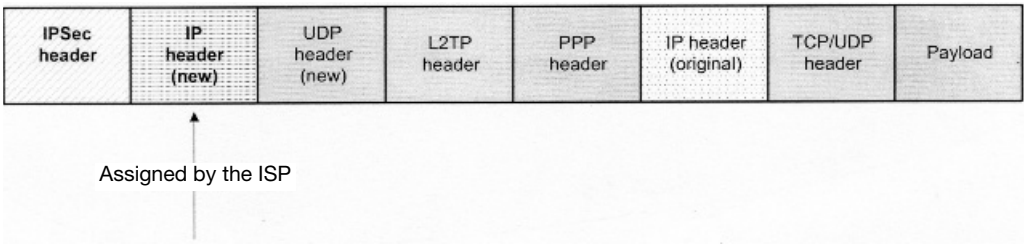

Figure 5.5 Encapsulation for secure network access with L2TP and IPSec.

Linux. As a consequence, S/WAN is no longer an active initiative. Details on going projects for Linux are available at http://www.freeswan.org.)

5.4.3 Application Security

Many security protocols operate at the application layer, which makes them independent of the lower layers. The whole gamut of security services is now available, namely:

1. Confidentiality, total or selective by field or by traffic flow
2. Data integrity
3. Peer entity authentication
4. Access control
5. Nonrepudiation of transmission with proof of origin
6. Nonrepudiation of reception with proof of reception

To illustrate, the Secure Shell (SSH[1]) provides security at the application layer; it allows a user to log on, execute commands, and transfer files securely. Thus, it can replace other applications, such as telnet, rlogin, rsh, rcp [35–37]. In reality, there are two distinct protocols SSH1 and SHH2. Both bind to the same TCP port. One important difference is that SSH2 has an explicit capability to secure ftp as well. Both are freely available specifications with freeware and commercial implementations. Guidelines for management of security with SSH are available [38].

In the rest of this chapter, we give an overview of the mechanisms used to implement security service. The objective is to present sufficient background for understanding the applications and not to give an exhaustive review. For a comprehensive discussion of the mathematics of cryptography and its applications, the reader can consult several excellent textbooks, such as Schneier [39] and Menezes et al. [40].

5.5 MESSAGE CONFIDENTIALITY

Confidentiality guarantees that information will be communicated solely to the parties that are authorized for its reception. Concealment is achieved with the help of encryption algorithms. There are two types of encryption: symmetric encryption, where the opera-

[1]Secure Shell and SSH are registered trademarks of SSH Communications Security, Ltd. of Finland.

tions of message obfuscation and revelation use the same secret key, and public key encryption, where the encryption key is secret and the revelation key is public.

5.5.1 Symmetric Cryptography

Symmetric cryptography is the tool employed in classic systems. The key that the sender of a secret message utilizes to encrypt the message is the same as the one that the legitimate receiver uses to decrypt the message. Obviously, key exchange among the partners has to occur before the communication, and this exchange takes place through other secured channels. Figure 5.6 illustrates the operation.

Let M be the message to encrypt with the encryption process E with a symmetric key K. The result will be the ciphertext C such that:

$$E[K(M)] = C$$

The decryption process D is the inverse function of E that restores the clear text:

$$D(C) = M$$

There are two main categories of symmetric encryption algorithms: block encryption algorithms and stream cipher algorithms. Block encryption acts by transforming a block of data of fixed size, generally 64 bits, in encrypted blocks of the same size. Stream ciphers convert the clear text one bit at a time by combining the stream of bits in the clear text, with the stream of bits from the encryption key using an exclusive OR (XOR).

Table 5.1 presents some algorithms for symmetric encryption commonly used in commercial applications.

The main drawback of symmetric cryptography systems is that both parties must obtain, one way or another, the unique encryption key. This is possible without too much trouble within a closed organization; on open networks, however, the exchange can be intercepted. Public key cryptography, which was proposed in 1976 by Diffie and Hellman, is one solution to the problem of key exchange [49].

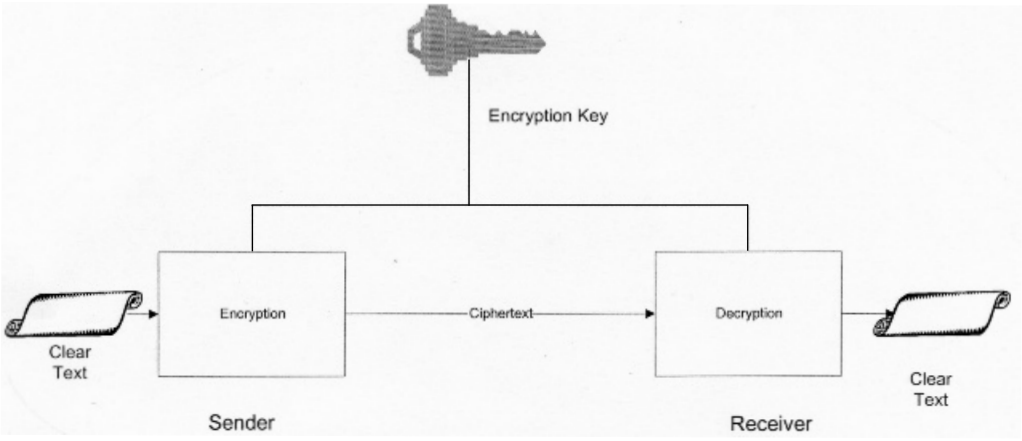

Figure 5.6 Symmetric encryption.

Table 5.1 Symmetric Encryption Algorithms in Commercial Applications

Algorithm	Name and Comments	Type of Encryption	Key Length in bits	Standard
AES	Advanced Encryption Standard	Blocks of 128, 192, or 256 bits	128, 192, or 256	FIPS 197
DES	Data Encryption Standard	Blocks of 64 bits	56	FIPS 81,1981 ANSI X3.92 [41], X3.105 [42], X3.106 [43], ISO 8372 [44], ISO/IEC 10116
IDEA (Lai and Massey [45, 46])	International Data Encryption Algorithm, apparently one of the best and most secure algorithms commercially available	Blocks of 64 bits	128	
RC2	Developed by Ronald Rivest [39, pp. 319–320]	Blocks of 64 bits	Variable, 40 bits for export from the United States	No, and proprietary
RC4	Developed by R. Rivest [39, pp. 397–398]	Stream	40 or 128	No, but posted on the Internet in 1994
RC5	Developed by R. Rivest [47]	Blocks of 32, 64, or 128 bits	Variable up to 2048 bits	No, and proprietary
SKIPJACK	An algorithm developed in the United States by the National Security Agency (NSA) for applications with the PCMCIA card Fortezza[a]	Blocks of 64 bits	80	Declassified algorithm whose Version 2.0 is available at http://csrc.nist.gov/encryption/skipjack-kea.htm.
Triple DES	Also called TDEA	Blocks of 64 bits	112	ANSI X9.52 [48]

[a]Fortezza is a Cryptographic Application Programming Interface (CAPI) that the NSA has defined for security applications on PCMCIA cards incorporating SKIPJACK.

5.5.2 Public Key Cryptography

Algorithms of public key cryptography introduce a pair of keys for each participant, a private key, SK, and a public key, PK. The keys are constructed in such a way that it is practically impossible to reconstitute the private key with the knowledge of the public key.

Consider two users, A and B, each having a pair of keys (PK_A, SK_A) and (PK_B, SK_B), respectively. Thus,

1. To send a secret message x to B, A encrypts it with B's public key and then transmits the encrypted message to B. This is represented by

$$e = PK_B(x)$$

2. B recovers the information using his or her private key SK_B. It should be noted that only B possesses SK_B, which can be used to identify B. The decryption operation can be represented by

$$x = SK_B(e) \qquad \text{or} \qquad x = SK_B[PK_B(x)]$$

3. B can respond to A by sending a new secret message x' encrypted with the public key PK_A of A:

$$e' = PK_A x'$$

4. A obtains x' by decrypting e':

$$x' = SK_B e' \qquad \text{or} \qquad x' = SK_A[PK_A\, x']$$

The diagram in Figure 5.7 summarizes these exchanges. Clearly, the public key is the encryption key and the private key is the recovery key.

It is worth noting that the preceding exchange can be used to verify the identify of each participant. More precisely, A and B are identified by the possession of the decryption key, SK_A or SK_B, respectively. A can determine if B possesses the private decryption key SK_B if the initial message x is included in the returned message x' that B sends. This indicates to A that the communication has been established with the entity that possesses SK_B. B can also confirm the identity of A in the same way.

The de facto standard for public key encryption is the algorithm RSA invented by Ronald Rivest, Adi Shamir, and Leonard Adleman in 1978 [50].

5.6 DATA INTEGRITY

The objective of the integrity service is to eliminate all possibility of nonauthorized modification of messages during their transit from the sender to the receiver. The traditional

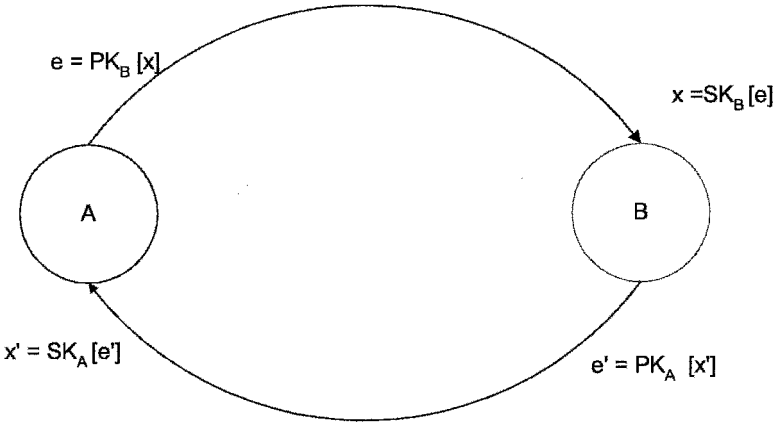

Figure 5.7 Confidentiality of messages with public key cryptography. ITU-T Recommendation X.509. (From the International Telecommunication Union, 2000. With permission.)

form to achieve this security is to stamp the letter envelope with the wax seal of the sender. Transporting this concept to electronic transactions, the seal will be a sequence of bits associated univocally with the document to be protected. This sequence of bits will constitute a unique and unfalsifiable "fingerprint" that will accompany the document sent to the destination. The receiver will then recalculate the value of the fingerprint from the received document and compare the value obtained with the value that was sent. Any difference will indicate that message integrity has been violated.

The fingerprint can be made to depend on the message content only by applying a hash function, on the message content and the sender's private key in the case of a public key encryption algorithm, or on the message content and a secret key that only the sender and the receiver know in the case of a symmetric encryption algorithm. In the first case, there are no secrets in the operation and anyone can calculate the fingerprint on the basis of the message and the hash function, provided that the hash function is known. In the second case, any person who has access to the public key of the sender and who knows the hash algorithm would be able to verify the message integrity. In the third case, only the receiver will be able to verify the integrity.

It should be noted that lack of integrity can be used to break confidentiality. For example, for some algorithms, attacks on the initialization vectors can be used to break down the confidentiality code.

5.6.1 Verification of the Integrity with a One-Way Hash Function

A *hash function* is a function that converts a sequence of characters of any length into a chain of characters of a fixed length, L, usually smaller than the original length, called a hash value. A *one-way hash function* is a function that can be calculated relatively easily in one direction, but with considerable difficulty in the inverse direction. A one-way hash function is sometimes called a compression function or a contraction function.

To verify the integrity of a message whose fingerprint has been calculated with the hash function $H(\ )$, this function should also be a one-way function; i.e., it should meet the following properties:

1. Absence of collisions, in other words, the probability of obtaining the same hash value with two different texts should be almost null. Thus, for a given message, x_1, the probability of finding a different message, x_2, such that $H(x_1) = H(x_2)$, is extremely small. For the collision probability to be negligible, the size of the hash value, L, should be sufficiently large.
2. Impossibility of inversion; given the fingerprint, h, of a message, x, it is practically impossible to calculate x such that $H(x) = h$.
3. A wide spread among the output values so that a small difference between two messages should yield a large difference between their fingerprints. Thus, any slight modification in the original text should, on average, affect half of the bits of the fingerprint.

Consider the message X. It will have been divided into n blocks, each consisting of B bits. If needed, padding bits would be appended to the message, according to a defined scheme, so that the length of each block reaches the necessary B bits. The operations for

cryptographic hashing are described using a compression function $f(\ )$ according to the following recursive relationship:

$$h_i = f(h_{i-1}, x_i), \qquad i = 1, \ldots, n$$

In this equation, h_0 is the vector that contains an initial value of L bits and $x = \{x_1, x_2, \ldots, x_n\}$ is the message subdivided into n vectors of B bits each. Some commonly used hash algorithms are listed in Table 5.2.

For MD5 and SHA-1, the message is divided into blocks of 512 bits. The padding consists in appending to the last block a binary "1," then as many "0" bits as necessary for the size of the last block, with padding to be 448 bits. Next, a suffix of 8 octets is added to contain the length of the initial message (before padding) coded over 64 bits, which brings the total size of the last block to 512 bits of 64 octets.

In 1994, two researchers, van Oorschot and Wiener, were able to detect collisions in the output of MD5 [56], which explains its gradual replacement with SHA-1. (*Note:* Many authors use SHA-1, SHA-1, and SHA interchangeably.)

Table 5.2 Some Commonly Utilized Hash Functions

Algorithm	Name	Length of the Fingerprint (L) in bits	Block Size (B) In bits	Standardization
DSMR	Digital Signature Scheme Giving Message Recovery			ISO/IEC 9796
MCCP	Banking key management by means of public key algorithms. Algorithms using the RSA cryptosystem. Signature construction by means of a separate signature			ISO/IEC 1116-2
MD4	Message Digest Algorithm	128	512	No, but described in RFC 1320 [51]
MD5	Message Digest Algorithm	128	512	No, but described in RFC 1321 [52]
RIPEMD	Extension of MD4, developed during the European project RIPE [40, p. 380]	128	512	
RIPEMD-128	Dedicated hash function #2	128	512	ISO/IEC 10118-3 [53]
RIPEMD-160	Improved version of RIPEMD [54]	160	512	
SHA	Secure Hash Algorithm (replaced by SHA-1)	160	512	FIPS 180
SHA-1	Dedicated Hash-Function #3 (revision and correction of the Secure Hash Algorithm)	160	512	ANSI X9.30:2 [55] ISO/IEC 10118-3 [53] FIPS 180-1

5.6.2 Verification of Integrity with Public Key Cryptography

An encryption algorithm with a public key is called *permutable* if the decryption and encryption operations can be inverted, i.e., if

$$M = PK_X[SK_X(M)]$$

In the case of encryption with a permutable public key algorithm, an information element, M, that is encrypted by the private key, SK_X, of an entity, X, can be read by any user possessing the corresponding public key, PK_X. A sender can therefore sign a document by encrypting it with a private key reserved for the signature operation to produce the seal that accompanies the message. Any person who knows the corresponding public key will be able to decipher the seal and verify that it corresponds to the received message.

Another way of producing the signature with public key cryptography is to encrypt the fingerprint of the document. This is because the encryption of a long document using a public key algorithm imposes substantial computations and introduces excessive delays. It is therefore beneficial to use a digest of the initial message before applying the encryption. This digest is produced by applying a one-way hash function to calculate the fingerprint, which is then encrypted with the sender's private key. At the destination, the receiver recomputes the fingerprint. With the public key of the sender, the receiver will be able to decrypt the fingerprint to verify if the received hash value is identical to the computed hash value. If both are identical, the signature is valid. (*Note:* The signature obtained in this way is sometimes called compression, contraction, message digest, fingerprint, cryptographic checksum or message integrity check (MIC) [38].)

The block diagram in Figure 5.8 represents verification of integrity with public key encryption. In this figure "h" represents the hash function, "C" the encryption function, and "D" the decryption function. The public key algorithms that are frequently used to calculate digital signatures are listed in Table 5.8.

Even though this message allows verification of message integrity, it does not guarantee that the identity of the sender is authentic. Therefore, prior to verifying the signature in a signed message, the sender and their parameters need to be authenticated. In the case of public key encryption of the hash value, authentication requires the use of certificates, as will be explained later. (*Note:* A signature produced from a message with the signer's private key and then verified with the signer's corresponding public key is sometimes called a signature scheme with appendix [58].)

5.6.3 Blind Signature

A *blind signature* is a special procedure for a notary to sign a message using the RSA algorithm for public key cryptography without revealing the content [59, 60]. One possible utilization of this technique is to time-stamp digital payments.

Consider a debtor who would like to have a payment blindly signed by a bank. The bank has a public key, e, a private key, d, and a public modulo, N. The debtor chooses a random number, k, between 1 and N, and keeps this number secret.

The payment p is "enveloped" by applying the following formula:

$$(p\ k^e) \bmod N$$

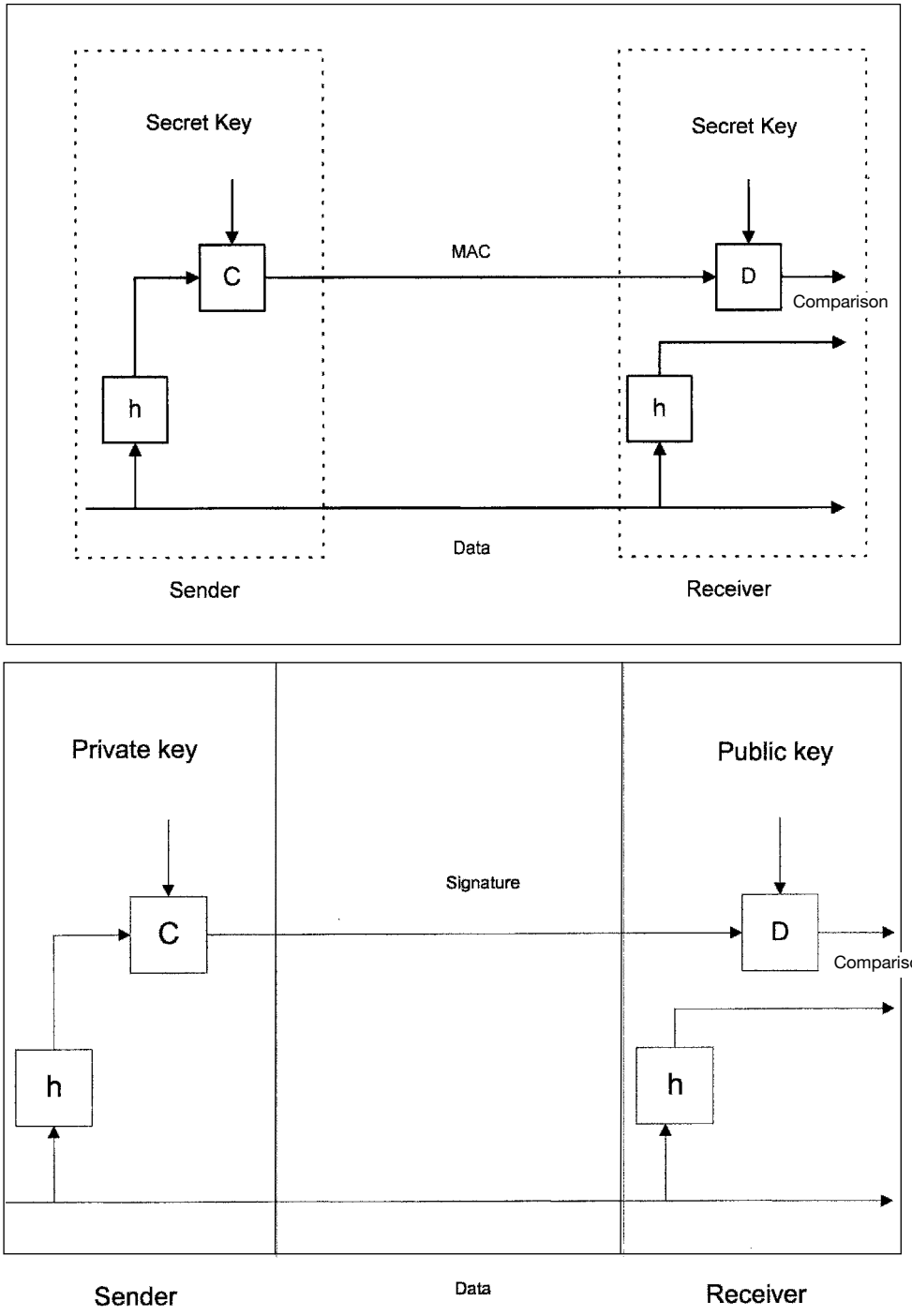

Figure 5.8 Computation of the digital signature using public key algorithms and hashing.

Table 5.3 Public Key Algorithms Used to Compute Digital Signatures

Algorithm	Comments	Length of the Fingerprint	Standard
DSA[a]	Digital Signature Algorithm, which is a variant of the ElGamal algorithm. It is a part of the Digital Signature Standard (DSS) that was proposed by the National Institute of Standards and Technology (NIST) in 1994	512 to 1024 bits	FIPS 186I ANSI X.9.30:1 [1]
ElGamal	Nondeterministic algorithm where a message corresponds to several signatures; it uses discrete logarithms [57]	Variable	—
RSA	This is the defacto standard algorithm for public key encryption; it can also be used to calculate signatures.	512 to 1024 bits	ISO/IEC 9796

[a]The United States federal government mandates the use of the DSA for signing electronic procurements.

before sending the message to the bank. The bank signs it with its private key so that

$$(p\ k^e)^d \bmod N = p^d\ k \bmod N$$

and returns the payment to the debtor. The debtor can now extract the signed note by dividing the number by k. To verify that the note received from the bank is the one that has been sent, the debtor can raise it to the e power because (as will be shown in Appendix II):

$$(p^d)^e \bmod N \equiv p \bmod N$$

The various payment protocols for digital money take advantage of blind signatures to satisfy the conditions of anonymity.

5.6.4 Verification of Integrity with Symmetric Cryptography

The message authentication code (MAC) is the result of a one-way hash function that depends on a secret key. This mechanism guarantees simultaneously the integrity of the message content and the authentication of the sender.

The most obvious way of constructing a MAC is to encrypt the hash value with a block-symmetric encryption algorithm. The MAC is then affixed to the initial message and the whole is sent to the receiver. The receiver recomputes the hash value by applying the same hash function on the received message and compares the result obtained with the decrypted MAC value. The equality of both results confirms data integrity.

The block diagram in Figure 5.9 depicts the operations where “h” represents the hash function, “C” the encryption function, and “D” the decryption function.

Another variant of this method is to append the secret key to the message that will be condensed with the hash functions.

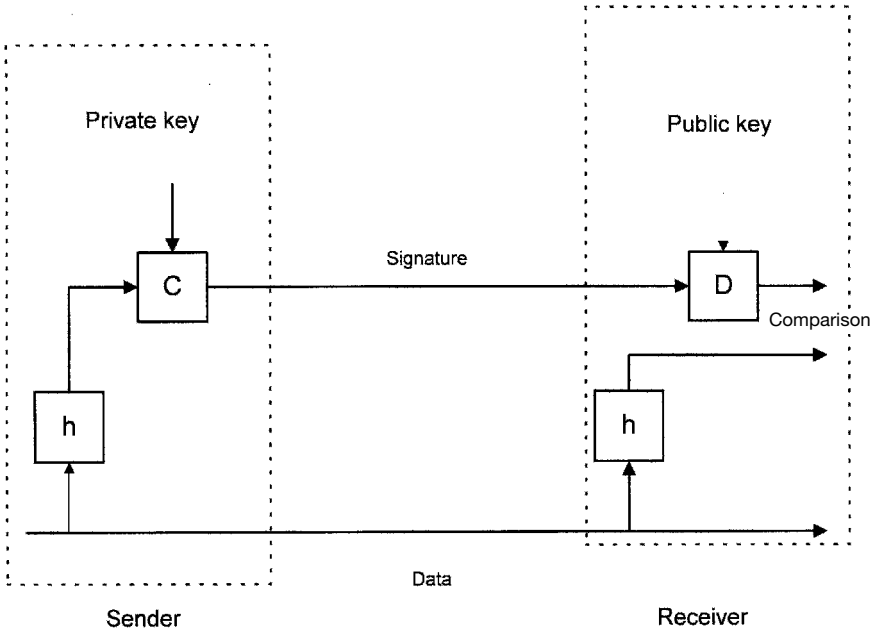

Figure 5.9 Digital signature with symmetric encryption algorithms.

It is also possible to perform the computations with the compression function $f(\ )$ and use as an initial value the vector of the secret key, k, of length L bits in the following recursion:

$$k_i = f(k_{i-1}, x_i), \qquad i = 1, \ldots, n$$

where $x = \{x_1, x_2, \ldots, x_n\}$ is the message subdivided into n vectors, each of B bits. The MAC is the value of the final output k_n. (*Note:* Some authors call the MAC the "integrity check value" or the "cryptographic checksum.")

The following keyed-hashing method augments the speed of computation in software implementation and increases the protection, even when the one-way hash algorithm experiences some rare collisions [59].

Consider the message X subdivided into n vectors of B bits each, and two keys (k_1 and k_2), each of L bits. The padding bits are added to the end of the initial message according to a determined pattern. The hashing operations can thus be described with the help of two compression functions $f_1(\ )$ and $f_2(\ )$:

$$k_i^1 = f_1(k_{i-1}^1, x_i), \qquad i = 1, \ldots, n$$

$$k_i^2 = f_2(k_{i-1}^2, k_i^1)$$

where k_0^1 and k_0^2 are the initial values of k_1 and k_2, respectively, and $x = x_1, x_2, \ldots, x_n$.

The result that this method yields is called the Nested Message Authentication Code (NMAC). It is in effect constructed by applying compression functions in sequence, the

first on the padded initial message and the second on the product of the first operation after padding.

The disadvantage of this method is that it requires access to the source code of the compression functions to change the initial values. In addition, it requires usage of two secret keys. This explains the current popularity of the Hashed Message Authentication Code (HMAC) described in RFC 2104 [62]. This method uses one single key k of L bits.

Assuming that the function $H(\ )$ represents the initial hash function, the value of the HMAC is computed in the following manner:

$$HMAC_k(x) = H[\bar{k} \oplus opad \| H(\bar{k} \oplus ipad, x)]$$

In this construction, $\bar{k}$ is the vector, k, of a minimum length of L bits, which after padding with a series of "0" bits, will reach a total length of B bits. The variables *opad* and *ipad* are constants for outer padding and inner padding, respectively. The variable *opad* is formed with the octet 0x36 repeated as many times as needed to constitute a block of B bits, while the variable *ipad* is the octet 0x5C repeated as many times as need. For MD5 and SHA-1 the number of repetitions is 64. Finally, the symbols || and $\oplus$ in the previous equation denote, respectively, the concatenation and exclusive OR operations.

It should be noted that with the following representation

$$k^1 = f_1(\bar{k} \oplus ipad)$$

$$k^2 = f_2(\bar{k} \oplus opad)$$

the HMAC becomes the same as the nested MAC. (*Notes:*

1. For the SSL protocol, the HMAC is denoted as MAC.
2. In IPSec, authentication and integrity checking are done simultaneously by using one of the keyed-hashing HMACs with MD5 or SHA-1.)

5.7 IDENTIFICATION OF PARTICIPANTS

Identification is the process of ascertaining the identity of a participant (whether a person or a machine) by relying on uniquely distinguishing feature. This contrasts with authentication, which is confirmation that the distinctive identifier indeed corresponds to the declared user.

Authentication and identification of a communicating entity take place simultaneously when that party proposes to the verifier in private a secret that is only shared between them, for example, a password or a secret encryption key. Another possibility is to pose a series of challenges that only the legitimate user is supposed to be capable of answering.

Digital signature is the usual means of identification because it associates a party (a user or a machine) with a shared secret. Other methods of simultaneous identification and authentication of human users exploit biometric characteristics such as fingerprints, voice prints, the shape of the retina, the form of the hand, etc. This is elaborated in the following section.

5.7.1 Biometric Identification

Biometric identification techniques, reserved until recently for military uses and law enforcement agencies, are being considered for user identification in civilian applications. The use of biological attributes for identification and authentication bypasses some of the problems associated with cryptography (e.g., key management). This explains the interest in biometrics in large-scale civilian applications such as mobile telephony, electronic commerce, or telework.

There are two main categories of biometric features. The first category relates to behavioral patterns and acquired skills such as speech, handwriting, or keystroke patterns. In contrast, the second category comprises physiological characteristics such as facial features, iris morphology, retinal texture, hand geometry, or fingerprints. Methods based on gait, odor, or genetic composition using DNA have limited applications for on-line systems.

The usage of biometric systems includes three steps: image acquisition during the registration phase, features extraction, and identification or verification. The digital image of the person under examination originates for a sensor in the computer peripheral (a microphone, for example). This image is processed to extract a compact profile that should be unique to that person. This profile or signature is then archived in a reference database that can be centralized or distributed according to the architecture of the system. In most of these cases, registration cannot be done on-line; rather the person has to be physically present in front of a registrar to record the necessary biometric template.

Biometric identification systems ascertain the identity of the end user by matching the biometric data with an entry in a database to supplement another identifier (password, badge, etc.). Verification systems, in contrast, match biometric data with what is stored in the user credential (e.g., a smart cart) to verify access privileges.

It should be noted that biometric systems are not foolproof. The accuracy of an identification system is measured in terms of the rate of mix-up of identities and the rate of rejects of authorized identities. In contrast, the performance of biometric verification systems is assessed in terms of rate of false rejects, i.e., the rejection of authorized identities and the rate of false acceptances. These rates are interdependent, and are adjusted according to the required level of security.

The choice of a particular systems depends on several factors:

1. Accuracy and reliability of the identification or verification. The result should not be affected by the environment or by aging.
2. Cost of installation, maintenance, and operation.
3. Scale of applicability of the technique; for example, handwriting recognition is not useful for illiterate people.
4. Ease of use.
5. Reproducibility of results. In general, physiological characteristics are more reproducible than behavioral characteristics.
6. Resistance to counterfeit and attacks.

Speaker identification technology verifies the end user by comparing a digitized sample of a person's voice with a stored vocal imprint. Although the system is easy to use, its effectiveness depends on the handset characteristics, the background noise, and the link type (whether terrestrial or radio). Furthermore, under some conditions (20 hours of selected material), there are algorithms that can mimic someone's speech.

In handwriting recognition systems, the user writes on a special pad with a pressure-sensitive pen. The dynamic movement of the pen is described by tens of parameters, such as the pressure exercised on the pad, the speed and direction of the movement, the accelerations and decelerations, and the angle of the letter. One limitation of handwriting recognition is that the rate of false rejects may be too high.

Keystroke recognition techniques measure an individual's typing patterns in terms of rhythm, speed, and duration and pressure of keystrokes. The method for parameter estimation requires several repetitions of a predetermined sequence of characters (for example, the log-in and the password).

Facial scans are used for on-line or off-line identification. In the latter case, a reference image, whose size ranges from 100 to 800 octets, is stored on a smart card in the user's possession.

The retina is a special tissue of the eye that responds to light pulses by generating proportional electrical discharges to the optical nerve. In retinal recognition systems, a map of the retinal blood vessels is recorded with the help of a charge-coupled device (CCD) using reflections from a low-intensity infrared source. The descriptor is a vector of 35 octets that is not only unique to the individual but also stable over time. The main drawback of the system is that the person has to look into the infrared ray source through a special eyepiece.

A less invasive technique is the description of the iris texture with a numeric code of 256 octets (2048 bits). The person to be identified needs merely to face a desktop camera at a distance of 1 m. The accuracy is very high and the error probability is of the order of 1 for 1.2 million. It is even possible to distinguish among identical twins and to separate the two irises of the same person. Iris recognition is now being evaluated in some U.S. airports to speed up passenger processing. Another potential application is the identification of users of automatic bank teller machines for the control of access either to a physical building or equipment or to network resources. The accuracy however, may be affected by contact lenses.

The traditional method for collecting fingerprints is to swipe the finger tips (or the palm) in a special ink and then press them over paper to record a negative image. This image is processed to extract user-specific information or *minutiae*. New imaging methods allow the capture of the fingerprints with optical, optoelectronic, electric, or thermal transducers. For example, variation in the capacitance between the user's fingers and sensors on the surface of a special mouse can help draw the contour of the fingerprint. Another technique relies on a low tension alternating current injected into the fingertips to measure the changes in the electric field between a resin plate and the derma. These variations reproduce the details of the fingerprint. Thermal techniques rely on a transducer to measure the temperature gradient on the mouse's surface. Finally, optoelectronic methods employ a layer of polymers to record the image of the fingerprint that a transducer converts into a proportional electric current. In these methods, each minutia takes about 16 octets on the average; therefore, the image size varies between 500 and 5000 octets, depending on the algorithm.

Recognition of hand geometry is already in use for access control in large-scale commercial applications. Some U.S. airports (e.g., New York and Newark) are using it to accelerate the admission of frequent travelers (those with more than five entries per year). The user positions the hand on a plate between a set of guiding pins. This plate is surrounded by mirrors on three sides to capture the hand sideways and from the top with a digital camera. The hand geometry is described using 90 characteristics in the form of a 9-

octets vector. These parameters are obtained by the averaging of several (3 to 5) shots. The time for taking one picture is about 1.2 s.

5.7.2 Summary and Evaluation

Table 5.4 gives the required memory for storing selected biometric identifiers [61, 62].

At this stage and regardless of the biometric technology, there is little commonality among the various methods being proposed and/or their implementations. In addition, there are no agreed upon protocols for measuring and comparing total system performance in terms of processing speed, reliability, and security of the hardware and software package. There is also a need for standardized protocols to assess a system's vulnerability to attacks or to compare performance in an operational environment. This lack of standards is hampering the large-scale acceptance of biometric identification. Users are concerned about the long-term viability of any solution they may select and the cost of switching methods or suppliers in the future. A related concern is that of being locked into a specific implementation or supplier. Application developers, in turn, are not sure what method deserves their full attention.

Awareness of these roadblocks has spurred standardization activities to facilitate data exchanges among various implementations irrespective of the biometric method. NIST and the Federal Bureau of Investigation (FBI) have made available a large database of fingerprints gathered from crime scenes and the corresponding minutiae. This database will help developers in their effort to train and evaluate new algorithms for fingerprint recognition. In 1995, the Committee on Security Policy Board established by President Clinton chartered the Biometric Consortium (BC) to be the focal point for the U.S. government on research, development, testing, evaluation, and application of biometric-based systems for personal identification/verification. The BC cosponsors activities at NIST and at San Jose State University in California.

The U.S. Department of Defense (DOD) initiated a program to develop a standard application interface called the *Human-Authentication–Application Program Interface* (HA-API) to decouple the software of the applications from the technology used to capture the biometric data. After publishing, in April 1998, Version 2.0 of this API, activities merged with those of the BioAPI Consortium (*http://www.bioapi.org*). This consortium groups hardware and software companies as well as supplier of biometric peripherals. In March 2000, the consortium published Version 1.0 of a BioAPI and reference realizations for Windows, UNIX, Linux, and Java. All of these implementations are in the public domain.

Table 5.4 Required Storage Memory for Biometrical Identifiers

Identifier	Required Memory (in octets)
Photo image	1000–1500
Voice print	1000–2000
Handwritten scan	500–1500
Face recognition	500–1000
Fingerprint	500–5000
Iris scan	256–512
Retinal scan	35
Hand geometry	9

The BioAPI specification was the basis of the INCITS 358, a standard that the Technical Committee M1 on Biometrics for the InterNational Committee for Information Technology Standards (INCITS) has developed as an American National Standards Institute (ANSI) Standard.

In parallel, efforts within the ANSI X9.F4 working group have resulted in a common format to exchange biometric data among various systems known as Common Biometric Exchange File Format (CBEFF). This is the format to be recognized by the BioAPI. It was agreed that the International Biometric Industry Association (IBIA) (*http://www.ibia.org*) will act as the registration authority for the formats to be recognized. Finally, in X9.84 [65], ANSI has defined a data object model that is compatible with CBEFF and is suitable for securing physical and remote access within the financial industry. The standard gives guidance on proper controls and procedures for using biometrics for identification and authentication .

Other standardization initiatives are pursued by the Association for Biometrics (*http://www.afb.org.uk*) in the United Kingdom, and the *Bundesamt für Sicherheit in der Informationtechnik* (BSI—Federal Information Security Agency) (*http://www.bsi.bund.de*) in Germany. Finally, joint work by ISO and IEC aims at a standard for personal verification through biometric methods with the use of integrated circuit cards (e.g., smart cards). Potential applications include driver licenses and travel documents. The standard will be issued as ISO/IEC 7816, Part 11.

5.8 AUTHENTICATION OF PARTICIPANTS

The purpose of authentication of participants is to reduce, if not eliminate, the risk that intruders might masquerade under legitimate appearances to pursue unauthorized operations.

As has been previously stated, when participants utilize a symmetric encryption algorithm, they are the only ones who share a secret key. As a consequence, utilization of this algorithm guarantees, in theory, the confidentiality of the messages, the correct identification of correspondents, and their authentication. The key distribution servers also act as authentication servers, and the good functioning of the system depends on the capability of all participants to protect the encryption key.

In contrast, when participants utilize a public key algorithm, a user is considered authentic when that user can prove that he or she holds the private key that corresponds with the public key that is attributed to the user. A certificate issued by a certification authority indicates that it certifies the association of the public key (and therefore the corresponding private key) with the recognized identity. In this manner, identification and authentication proceed in two different ways, identity with the digital signature and authentication with a certificate. Without such a guarantee, a hostile user could create a pair of private/public keys and then distribute the public key as if it were that of the legitimate user.

Although the same public key of a participant could equally serve to encrypt the message that is addressed to that participant (confidentiality service) and to verify the electronic signature of the documents that the participant transmits (integrity and identification services), in practice a different public key is used for each set of services.

According to the authentication framework defined by ITU-T Recommendations X.500 and X.811 [66, 67], simple authentication can be achieved by one of several means:

1. Name and password in the clear.
2. Name, password, and a random number or a time stamp, with integrity verification through a hash function.
3. Name, password, a random number, and a time stamp, with integrity verification using a hash function.

Strong authentication requires a certification infrastructure that includes the following entities:

1. *Certification authorities* to back the users' public keys with "sealed" certificates (i.e., signed with the private key of the certification authority) after verification of the physical identity of the owner of each public key.
2. A database of authentication data (*directory*) that contains all the data relative to the private encryption keys, such as their value, the duration of validity, and the identity of the owners. Any user should be able to query such a database to obtain the public key of the correspondent or to verify the validity of the certificate that the correspondent would present.
3. A *naming* or *registering authority*. This authority can be distinct from the certification authority, and its principal role is to define and assign unique *distinguished names* to the different participants.

The certificate guarantees correspondence between a given public key and the entity whose unique distinguished name is contained in the certificate. This certificate is sealed with the private key of the certification authority. When the certificate owner signs documents with the private signature key, the partners can verify the validity of the signature with the help of the corresponding public key that is contained in the certificate. Similarly, to send a confidential message to a certified entity, it is sufficient to query the directory for the public key of that entity and then use that key to encrypt messages that only the holder of the associated private key would be able to decipher.

5.9 ACCESS CONTROL

Access control is the process by which only authorized entities are allowed access to the resources as defined in the access control policy. It is used to counter the threat of unauthorized operations such as unauthorized use, disclosure, modification, destruction of protected data or denial of service to legitimate users. ITU-T Recommendation X.812 [68] defines the framework for access control in open networks. Accordingly, access control can be exercised with the help of a supporting authentication mechanism at one or more the following layers: the network layer, the transport layer, or the application layer. Depending on the layer, the corresponding authentication credentials can be X.509 certificates, Kerberos tickets, simple identity, and password pairs, etc.

There are two types of access control mechanisms: identity-based and role-based. Identity-based access control uses the authenticated identity of an entity to determine and enforce its access rights. In contrast, for role-based access control, access privileges depend on the job function and its context. Thus, additional factors may be considered in the definition of the access policy, for example, the strength of the encryption algorithm, the

type of operation requested, or the time of day. Thus, role-based access control provides an indirect means of bestowal of privileges through three distinct phases: the definition of roles, the assignment of privileges to roles, and the distribution of roles among users. This facilitates the maintenance of access control policies because it is sufficient to change the definition of roles to allow global updates without revising the distribution from top to bottom.

At the network layer, access control in IP networks is based on packet filtering using the protocol information in the packet header, specifically the source and destination IP addresses, and the source and destination port numbers. Access control is achieved through "line interruption" by a certified intermediary or a firewall, that intercepts and examines all exchanges before allowing them to proceed. The intermediary is thus a trusted third party that is located between the client and the server, as indicated in Figure 5.10. Furthermore, the firewall can be charged with other security services, such as encrypting the traffic for confidentiality at the network level or integrity verification using digital signatures. It can also inspect incoming and outgoing exchanges before forwarding them to enforce the security policies of a given administrative domain. However, the intervention of the trusted third party must be transparent to the client.

The success of packet filtering is vulnerable to packet spoofing if the address information is not protected and if individual packets are treated independently of other packets of the same flow. As a remedy, the firewall can include a proxy server or an application-level gateway that implements a subset of application-specific functions. The proxy is capable of inspecting all packets in light of previous exchanges of the same flow before allowing their passage in accordance to the security policy in place. Thus, by filtering incoming and outgoing electronic mail, file transfers, exchanges of Web applications, etc., application gateways can block nonauthorized operations and protect against malicious codes such as viruses. This is called a *stateful* inspection.

A third approach is to centralize management of the access control for a large number of clients and users with different privileges with a dedicated server. Several protocols have been defined to regulate the exchanges among network elements and access control servers. RFC 2865 [69] specifies Remote Authentication Dial in User Service (RADIUS) for client authentication, authorization and for collecting accounting information of the calls. In RFC 1492 [70] Cisco has described a protocol called Terminal Access Controller Access System (TACACS) which was later updated in TACACS+. Both RADIUS and TACACS+ require a secrete key between each network element and the server. Figure 5.11 depicts the operation of RADIUS in terms of a client/server architecture. The RADIUS client resides within the access control server while the server relies on an X.509 directory through the LDAP. Both X.509 and LDAP will be presented later in this chapter.

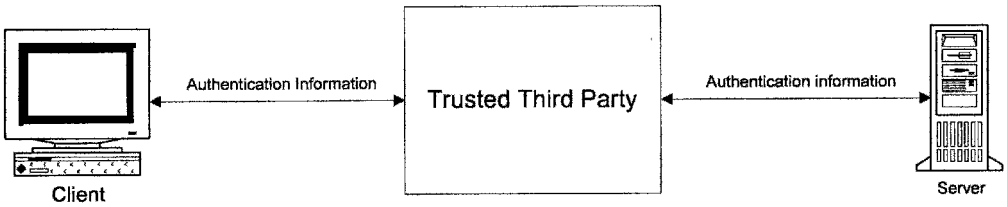

Figure 5.10 Authentication by line interruption at the network layer.

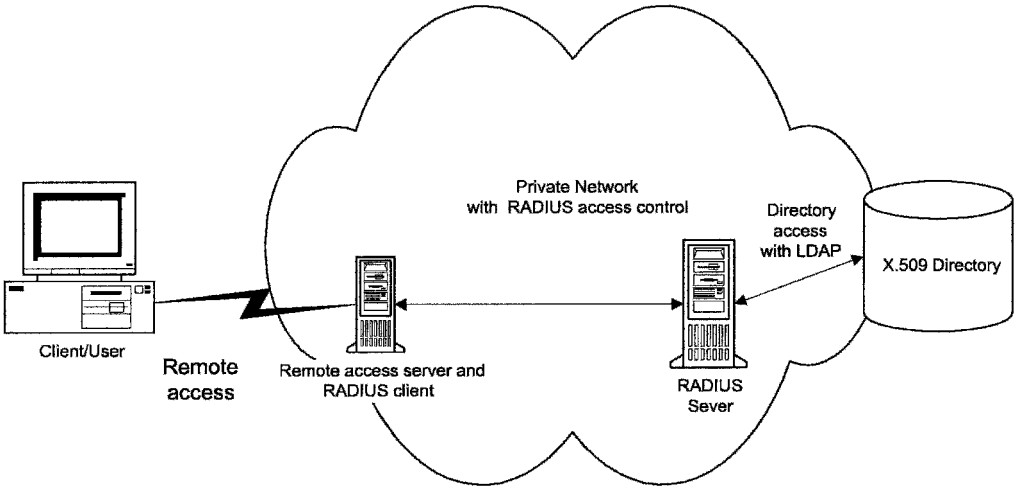

Figure 5.11 Remote access control with RADIUS.

Note that both server-to-client authentication and user-to-client authentication are outside the scope of RADIUS. Also, because, RADIUS does not include provisions for congestion control, large networks can suffer degraded performance and data loss.

Commercial systems implement two basic approaches for end user authentication: one-time password and challenge-response [16]. In a typical one-time password system, each user has a device that generates a number periodically (usually every minute) using the current time, the card serial number, and a secret key held in the device. The generated number is the user's one-time password. This procedure requires that the time reference of the access control server be synchronized with the card so that the server can regenerate an identical number.

In challenge-response systems, the user enters a personal identification number to activate hand-held authenticators (HHA), and then to initiate a connection to an access control server. The access control server, in turn, provides the user with a random number (a challenge), and the user enters this number into a hand-held device to generate a unique response. This response depends on both the challenge and some secret key shared between the user's device and the server. It is returned to the access control server to compare with the expected response and decide accordingly.

It should be noted that there are some known vulnerabilities in RADIUS or in its implementations [71].

5.9.1 Denial-of-Service

Denial-of-service attacks prevent normal network usage by blocking access of legitimate users to the network resources they are entitled to, by overwhelming the hosts with additional or superfluous tasks to prevent them from responding to legitimate requests or to slow their response time below satisfactory limits.

In a sense, denial-of-service results from failure of access control. Nevertheless, these attacks are inherently associated with IP networks for two reasons: (1) network control

data and user data share the same physical and logical bandwidths, and (2) IP is a connectionless protocol where the concept of admission control does not apply. As a consequence, when the network size exceeds a few hundred nodes, network control traffic (due, for example, to the exchange of routing tables) may, under some circumstances occupy a significant portion of the available bandwidth. Further, inopportune or ill-intentioned user packets may be able to bring down a network element (e.g., a router) thereby affecting not only all end points that rely on this network element for connectivity, but also all other network elements that depend on it to update their view of network status. Finally, in distributed denial of service attacks (DDOS), a sufficient number of compromised hosts may send useless packets towards a victim around the same time, thereby affecting the victim's resources or bandwidth or both [72, 73].

As a point of comparison, the current public switched telephone network uses an architecture called common channel signaling (CCS) whereby user data and network control data use totally separate networks and facilities. It is worth noting that CCS was introduced to protect against fraud. In the old architecture, called channel-associated signaling (CAS), the network data and the user data used separate logical channels, on the same physical support. Similarly, experience has shown that ATM can be exposed to the same risks of interruption because user traffic and network control messages share the same facilities even though they are virtually distinct [74].

Let us illustrate the preceding discussion with a few examples of denial-of-service attacks using several protocols of the IP stack: TCP, Internet Control Message Protocol (ICMP), and HTTP.

- The SYN flooding attack, one of the best known mechanisms of denial-of-service, perturbs the functioning of the TCP protocol [75]. It is well known that the handshake in TCP is a three-way exchange: a connection request with the SYN packet, an acknowledgment of that request with SYN/ACK packet, and finally a confirmation from the first party with the ACK packet [76, p. 216]. Unfortunately, the handshake imposes asymmetric memory and computational loads on the two endpoints, the destination being required to allocate large amounts of memory without authenticating the initial request. Thus, an attacker can paralyze the target machine, exhausting its available resources by sending a massive number of fake SYN packets. These packets will have spoofed source addresses, so the acknowledgments are sent to hosts that the victim cannot reach or that do not exist. Otherwise, the attack may fail, because unsolicited SYN/ACK packets at accessible hosts provoke the transmission of RST packets, which upon arrival, would allow the victim to release the resources allocated for a connection attempt.
- ICMP is a protocol for any arbitrary machine to communicate control and error information back to the presumed source. Thus, an ICMP echo request, or "ping," with the victim's address falsely indicated as the source and sent to all the machines of a given network using the subnet broadcast address can flood the victim with echo replies that will overwhelm its capacities.
- The Code Red worm exploits defects in the response of some Web server to an HTTP GET request larger than the regular size (a payload of 62 octets instead of 60 octets). Under specific conditions, the buffer overflow causes an upsurge in HTTP traffic and the infection of neighboring machines, which increases network traffic, thereby causing a massive disruption [77].

Given that IP does not separate user traffic from that of the network, the best solution is to identify all with trusted certificates. However, authentication of all exchanges increases the computational load, which may be excessive in commercial applications, as the lack of success of the protocol for payments with bankcard SET has shown. Short of this, defense mechanisms will be developed on a case-by-case basis to address specific problem as they arise, for example, resource exhaustion due to the SYN attack can be alleviated by limiting the number of concurrent pending TCP connections, reducing the time out for the arrival of the ACK packet before calling off the connection establishment, blocking packets to the outside that have source addresses from outside.

Another approach is to reequilibrate the computational load among the two parties by asking the requesting client to solve a *puzzle* in the form of simple cryptographic problems before the allocated resources needed to establish a connection. To avoid replay attacks, these problems are formulated using the current time, a server secret, and additional information from the client request [78]. This approach, however, requires programs for solving puzzles specific to each application: TCP, SSL, etc, which are incorporated in the client browser.

5.10 NONREPUDIATION

Nonrepudiation is a service that prevents a person who has accomplished an act from denying it later, in part or as a whole. Nonrepudiation is a legal concept to be defined through legislation. The role of informatics is to supply the necessary technical means to support the service offer according to the law. The building blocks of nonrepudiation include the electronic signature of documents, the intervention of a third party as a witness, time-stamping, and sequence numbers. Among the mechanisms for nonrepudiation are a security token sealed with the secret key of the verifier that accompanies the transaction record, time-stamping, and sequence numbers. Depending on the system design, the security token sealed with the verifier's secret key can be stored in a tamper-resistant cryptographic module. The generation and verification of the evidence often require the intervention of one or more entities external to parties to the transaction, such as a notary, a verifier, and an adjudicator of disputes.

ITU-T Recommendation X.813 [79] defines a general framework for nonrepudiation in open systems. Accordingly, the service comprises the following measures:

- Generation of the evidence
- Recording of the evidence
- Verification of the evidence generated
- Retrieval and reverification of the evidence

There are two types of nonrepudiation services:

1. *Nonrepudiation at the Origin* This service protects the receiver by preventing the sender from denying having sent the message.
2. *Nonrepudiation at the Destination* This service plays the inverse role of the preceding function. It protects the sender by demonstrating that the addressee has received the message.

Threats to nonrepudiation include compromise of keys or unauthorized modification or destruction of evidence. In public key cryptography, each user is the sole and unique owner of the private key. Thus, unless the whole system has been penetrated, a given user cannot repudiate the messages that are accompanied by his or her electronic signature. In contrast, nonrepudiation is not readily achieved in systems that use symmetric cryptography. A user can deny having sent the message by alleging that the receiver has compromised the shared secret or that the key distribution server has been successfully attacked. A trusted third party would have to verify each transaction to be able to testify in cases of contention.

Nonrepudiation at the destination can be obtained using the same mechanisms, but in the reverse direction.

5.10.1 Time-Stamping and Sequence Numbers

Time-stamping of messages establishes a link between each message and the date of its transmission. This permits the tracing of exchanges and prevents attacks by replaying old messages. If clock synchronization of both parties is difficult, a trusted third party can intervene as a notary and use its own clock as reference.

Intervention of the "notary" can be either of the following:

- Off-line to fulfill functions such as certification, key distribution, and verification, if required, without intervening in the transaction.
- On-line as an intermediary in the exchanges or as an observer collecting the evidence that might be required to resolve contentions. This is a similar role to that of a trusted third party of the network layer (firewall) or at the application layer (proxy), but with a different set of responsibilities.

Let us assume that a trusted third party combines the functions of the notary, the verifier, and the adjudicator. Each entity encrypts its messages with the secret key that has been established with the trusted third party before sending the message. The trusted third party decrypts the message with the help of this shared secret with the intervening party, time-stamps it, and then reencrypts it with the key shared with the other party. This approach requires the establishment of a secret key between each entity and the trusted third party that acts as a delivery messenger. Notice, however, that the time-stamping procedures have not been normalized and each system has its own protocol.

Detection of duplication, replay, as well as the addition, suppression, or loss of messages is achieved with the use of a sequence number before encryption. Another mechanism is to add a random number to the message before encryption. All these means give the addressee the ability to verify that the exchanges genuinely took place during the time interval that the time stamp defines.

5.11 SECURE MANAGEMENT OF CRYPTOGRAPHIC KEYS

Key management is a process that continues throughout the life cycle of the keys to thwart unauthorized disclosures, modifications, substitutions, reuse of revoked or expired keys, or unauthorized use. Security at this level is a recursive problem, because the same securi-

ty properties that are required in the cryptographic system must be satisfied in turn by the key management system.

The secure management of cryptographic keys relates to key production, storage, distribution, utilization, withdrawal from circulation, deletion, and archiving [80].

5.11.1 Production and Storage

Key production must be done in a random manner and at regular intervals depending on the degree of security required.

Protection of the stored keys has a physical aspect and a logical aspect. Physical protection consists of storing the keys in safes or in secured buildings with controlled access, whereas logical protection is achieved with encryption.

In the case of symmetric encryption algorithms, only the secret key is stored. For public key algorithms, storage encompasses the user's private and public keys, the user's certificate, and a copy of the public key of the certification authority. The certificates and the keys can be stored on the hard disk of the certification authority, but there is some risk of possible attacks or of loss due to hardware failure. In cases of microprocessor cards, the information related to security, such as the certificate and the keys, is inserted during card personalization. Access to this information is then controlled with a confidential code.

5.11.2 Distribution

The security policy defines the manner in which keys are distributed to entitled entities. Manual distribution by mail or special dispatch (sealed envelopes, tamper-resistant module) is a slow and costly operation that should only be used for distribution of the root key of the system. This is the key that the key distributor utilizes to send each participant their key.

An automatic key distribution system must satisfy all the criteria of security, in particular:

- Confidentiality.
- Identification of the participant.
- Data integrity by giving proof that the key has not been altered during transmission or that it was not replaced by a fake key.
- Authentication of the participants.
- Nonrepudiation.

Automatic distribution can be either point-to-point or point-to-multipoint. The Diffie–Hellman key exchange method [49] allows the two partners to construct a master key from two large numbers that have been previously exchanged in the clear. A symmetric session key is derived next, either by using this master key directly or from additional exchanges encrypted with this master key.

To distribute keys to several customers, an authentication server can also play the role of a trusted third party and distribute the secret keys to the different parties. These keys will be used to protect the confidentiality of the messages carrying the information on the key pairs.

5.11.3 Utilization, Withdrawal, and Replacement

The unauthorized duplication of a legitimate key is a threat to the security of key distribution. To prevent this type of attack, a unique parameter can be concatenated to the key, such as a time stamp or a sequence number that increases monotonically (up to a certain modulo).

The risk that a key is compromised increases proportionately with time and with usage. Therefore, keys have to be replaced regularly without causing service interruption. A common solution that does not impose a significant load is to distribute the session keys on the same communication channels used for user data. For example, in the SSL protocol, the initial exchanges provide the necessary elements to form keys that would be valid throughout the session at hand. These elements flow encrypted with a secondary key, called a key encryption key, to keep their confidentiality.

Key distribution services have the authority to revoke a key before its date of expiration after a key loss or because of the user's misbehavior.

5.11.4 Key Revocation

All user certificates must be revoked without delay for the following conditions: the user loses the right to employ a private key, if this key is accidentally revealed, or, more seriously, if the private key of a certification authority has been broken. Furthermore, these revocations have to be communicated to all the verifying entities in the shortest possible time. Similarly, the use of the revoked key by a hostile user should not be allowed. Nevertheless, the user will not be able to repudiate all the documents already signed and sent before the revocation of the key pair.

5.11.5 Deletion, Backup, and Archiving

Key deletion implies the destruction of all memory registers as well as magnetic or optical media that contain either the key or the elements needed for its reconstruction.

Backup applies only to encryption keys, and not to signature keys; otherwise, the entire structure for nonrepudiation would be put into question.

The keys utilized for nonrepudiation services must be preserved in secure archives to accommodate legal delays that may extend for up to 30 years. These keys must be easily recoverable in case of need, for example, in response to a court order. This means that the storage applications must include mechanisms to prevent unrecoverable errors from affecting the ciphertext.

5.11.6 Comparison Between Symmetric and Public Key Cryptography

Systems based on symmetric key algorithms pose the problem of ensuring the confidentiality of key distribution. This translates into the use of a separate secure distribution channel that is preestablished between the participants. Furthermore, each entity must have as many keys as the number of participants with whom it will enter into contact. Clearly, management of symmetric keys increases exponentially with the number of participants.

Public key algorithms avoid such difficulties, because each entity owns only one pair of private and public keys. Unfortunately, the computations for public key procedures are more intense than those for symmetric cryptography. The use of public key cryptography to ensure confidentiality is only possible when the messages are short, even though data

compression before encryption with the public key often succeeds in speeding up the computations. Thus, public key cryptography can complement symmetric cryptography to ensure the safe distribution of the secret key, particularly when safer means such as direct encounter of the participants, or the intervention of a trusted third party, are not feasible. Thus, a new symmetric key could be distributed at the start of each new session and, in extreme cases, at the start of each new exchange.

5.12 EXCHANGE OF SECRET KEYS: KERBEROS

Kerberos is a distributed system for on-line identification and authentication as well as access control using symmetric cryptography [81]. It is widely used for remote access to resources in a university computing center (files, printers, etc.) from nonsecure machines. Kerberos is now the default authentication option in Windows 2000.

The development of Kerberos started in 1978 within the Athena project at the Massachusetts Institute of Technology (MIT), financed by Digital Equipment Corporation (DEC) and IBM. Version 5 of Kerberos, which was published in 1994, is the version currently in use .

The system is built around a Kerberos key distribution center that enjoys the total trust of all participants with whom they all have already established symmetric encryption keys. Symmetric keys are attributed to individual users for each of their accounts when they register in person.

The key distribution center consists of an authentication server (AS) and a ticket-granting server (TGS) . The AS controls access to the TGS, which in turns controls access to specific resources. Every server shares a secret key with every other server. The algorithm used for symmetric encryption is the Data Encryption Standard (DES). Finally, during the registration of the users in person, a secret key is established with the AS for each user's account. With this arrangement, a client has access to multiple resources during a session with one successful authentication, instead of repeating the authentication process for each resource. The operation is explained below.

After identifying the end user with the help of a log-in and password pair, the AS sends the client a session symmetric encryption key to encrypt data exchanges between the client and the TGS. The session key is encrypted with the symmetric encryption key shared between the user and the AS. The key is also contained in the session ticket that is encrypted with the key preestablished between the TGS and the AS.

The session ticket, also called a ticket-granting ticket, is valid for a short period, typically a few hours. During this period, it can be used to request access to a specific service; this is why it is also called an initial ticket.

The client presents the TGS with two items of identification: the session ticket and an authentication title that is encrypted with the session key. The TGS compares the data in both items to verify the client authenticity and its access privileges before granting access to the specific server requested.

Figure 5.12 depicts the interactions among the four entities: (1) the client, (2) the AS, (3) the TGS, and (4) the desired server or resource, S.

The exchanges are now explained.

Message (1): Request of a Session Ticket A client, C, that desires to access a specific server, S, first requests an entrance ticket to the session from the Kerberos authentication server, AS. To do so, the client sends a message consisting of an identifier

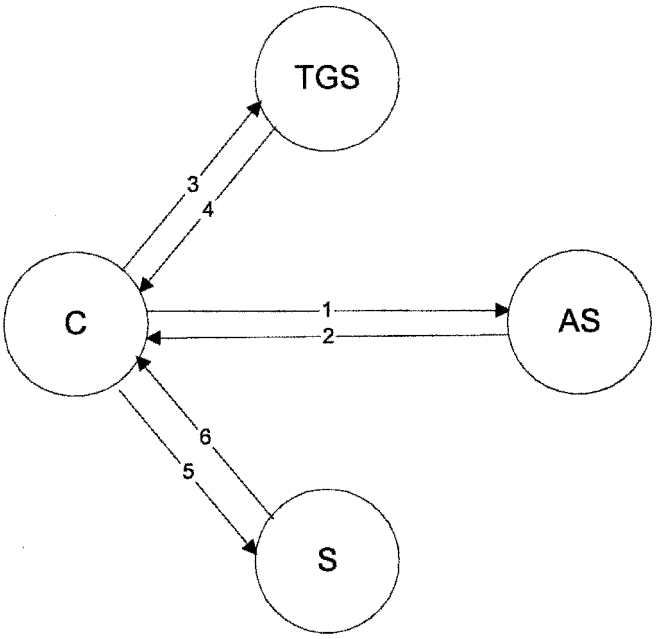

Figure 5.12 Authentication and access control in Kerberos.

(for example, a log-in and a password), the identifier of S, a time stamp, H_1, as well as a random number, Rnd, both to prevent replay attacks.

Message (2): Acquisition of a Session Ticket The Kerberos authentication server responds by sending a message formed of two parts: (1) a session key, K_{CTGS}, and the number, Rnd, that was in the first message, both coded with the client's secret key, K_C and (2) the session ticket, T_{CTGS}, destined for the TGS and encrypted by the latter's secret key between itself and the Kerberos authentication server.

The session (ticket-granting ticket) includes several pieces of information, such as the client name, C, its network address, Ad_C, the time stamp, H_1, the period of validity of the ticket, Val, and the session key, K_{CTGS}. All these items, with the exception of the server identity, TGS, are encrypted with the long-term key, K_{TGS}, that the TGS shares with the AS. Thus,

$$T_{CTGS} = \{TGS, K_{TGS}(C, Ad_C, H_1, Val, K_{CTGS})\}$$

and the message sent to the client is

$$K_c\{K_{CTGS}, Rnd\}, \{T_{CTGS}\}$$

where $K\{x\}$ indicates encryption of the message, x, with the shared secret key, K. The client decrypts the message with its secret key, K_C, to recover the session key, K_{CTGS}, and the random number. The client verifies that the random number received

is the same as that sent as a protection from replay attacks. The time stamp, H_1, is also used to protect against replay attacks. Although the client will not be able to read the session ticket because it is encrypted with K_{TGS}, it can extract it and relay it to the server.

By default the session ticket, T_{CTGS}, is valid for 8 hours. During this time, the client can obtain several service tickets for different services without the need for a new authentication.

Message (3): Request of a Service Ticket The client constructs an authentication title, Auth, that contains its identity, C, its network address, Ad_c, the service requested, S, a new time stamp, H_2, and another random number, Rnd_2, and then encrypts it with the session key, K_{CTGS}. The encrypted authentication title can be represented in the following form:

$$\text{Auth} = K_{CTGS}(C, Ad_C, S, H_2, Rnd_2)$$

The request of the service ticket consists of the encrypted authentication title and the session ticket, T_{CTGS}:

$$\text{Service request} = \{\text{Auth}, T_{CTGS}\}$$

Message (4): Acquisition of the Service Ticket The TGS decrypts the ticket content with its secret key, K_{TGS}, deduces the shared session key, K_{CTGS}, and extracts the data related to the client's service request. With knowledge of the session key, the server can decrypt the authentication title and compare the data in it with those that the client has supplied. This comparison gives formal proof that the client is the entity that was given the session ticket by the server. The time stamps confirm that the message was not an old message that has been replayed. Next, the TGS returns a service ticket for accessing the specific server, S.

The exchanges described by messages (3) and (4) can be repeated for all other servers available to the user, as long as the validity of the session ticket has not expired.

The message from the TGS has two parts: (1) a service key, K_{CS}, between the client and the server, S, and the number, Rnd_2, both coded with shared secret key, K_{CTGS}, and (2) the service ticket, T_{CS}, destined to the server, S, and encrypted by secret key, K_{STGS}, shared between the server, S, and the TGS.

As before, the service ticket destined for the server, S, includes several pieces of information, such as the identity of the server, S, the client's name, C, its network address, Ad_C, a time stamp, H_3, the period of validity of the ticket, Val, and, if confidentiality is desired, a service key, K_{CS}. All these items, with the exception of the server identity, S, are encrypted with the long-term key, K_{STGS}, that the ticket TGS shares with the specific server. Thus,

$$T_{CS} = \{S, K_{STGS}(C, Ad_C, H_3, \text{Val}, K_{CS})\}$$

and the message sent to the client is

$$K_{CTGS}\{K_{CS}, Rnd_2\}, \{T_{CS}\}$$

The client decrypts the message with the shared secret key, K_{CTGS}, to recover the service key, K_{CS}, and the random number. The client verifies that the random number received is the same as was sent as a protection from replay attacks.

Message (5): Service Request The client constructs a new authentication title, $Auth_2$, that contains its identity, C, its network address, Ad_c, a new time stamp, H_3, and another random number, Rnd_3, and then encrypts it with the service key, K_{CS}. The encrypted authentication title can be represented as the follows.

$$Auth_2 = K_{CS}(C, Ad_C, H_4, Rnd_3)$$

The request of the service consists of the encrypted new authentication title and the service ticket, T_{CS}:

$$\text{Service request} = \{Auth_2, T_{CS}\}$$

Message (6): Optional Response of the Server The server decrypts the content of the service ticket with the key, K_{STGS}, it shares with the TGS to derive the service key, K_{CS}, and the data related to the client. With knowledge of the service key, the server can verify the authenticity of the client. The time stamps confirm that the message is not a replay of old messages. If the client has requested the server to authenticate itself, it will return the random number, Rnd_3, encrypted by the service key, K_{CS}. Without knowledge of the secret key, K_{CS}, the server would have not have been able to extract the service key, K_{CS}.

The preceding description shows that Kerberos is mostly suitable for networks administered by a single administrative entity. In particular, the Kerberos key distribution center fulfills the following roles:

- It maintains a database of all secret keys (except of the key between the client and the server, K_{CS}). These keys have a long lifetime.
- It keeps a record of users' log-in identities, passwords, and access privileges. To fulfill this role, it may need access to an X.509 directory.
- It produces and distributes encryption keys and ticket-granting tickets to be used for a session.

5.12.1 Public Key Kerberos

The utilization of a central depot for all symmetric keys increases the potential of traffic congestion due to the simultaneous arrival of many requests. In addition, centralization threatens the whole security infrastructure, because a successful penetration of the storage could put all keys in danger [82]. Finally, management of the symmetric keys (distribution and update) becomes a formidable task when the number of users increases.

The public key version of Kerberos simplifies key management, because the server authenticates the client directly using the session ticket and the client's certificate sealed by the Kerberos certification authority. The session ticket itself is sealed with the client's private key and then encrypted with the server public key. Thus, the service request to the server can be described as follows:

$$\text{Service request} = S, PK_S \{Tauth, Kr, Auth\}$$

with

$$Auth = C, \text{certificate}, [Kr, S, PK_C, Tauth]SK_c$$

where Tauth is the initial time for authentication, Kr is a one-time random number that the server will use as a symmetric key to encrypt its answer, {. . .} represents encryption with the server public key, PK_S, while [. . .] represents the seal computed with the client's private key, SK_C. This architecture improves speed and security.

The operations of public key Kerberos are described in IETF RFC 1510 [83]. The official Web page for Kerberos is located at: http://web.mit.edu/kerberos/www/index.html. A frequently asked questions (FAQ) file on Kerberos can be consulted at the following address: ftp://athena-dist.mit.edu/pub/kerberos/KERBEROS.FAQ. Tung [84] contains a good compendium of information on Kerberos.

The Swedish Institute of Computer Science is distributing a free version of Kerberos, called Heidmal. This version was written by Johan Danielsson and Assar Westerlund, and includes improvements in security protocols, such as the support of Triple DES. A commercial version is TrustBroker available from CyberSafe at http://www.cybersafe.com.

5.13 EXCHANGE OF PUBLIC KEYS

5.13.1 Diffie–Hellman Exchange

The Diffie–Hellman algorithm, published in 1976, is the first algorithm for key exchange in public key algorithms. It exploits the difficulty in calculating discrete algorithms in a finite field, as compared with the calculation of exponentials in the same field.

The key exchange comprises the following steps:

1. The two parties agree on two random large integers, n and g, such that g is a prime with respect to n. These two numbers do not necessarily have to be hidden, but their choice can have a substantial impact on the strength of the security achieved.
2. A chooses a large random integer, x, and sends B the result of the computation:

$$X = g^x \bmod n$$

3. B chooses another large random integer, y, and sends to A the result of the computation:

$$Y = g^y \bmod n$$

4. A computes

$$k = Y^x \bmod n = g^{xy} \bmod n$$

5. Similarly, B computes

$$k = Y^x \bmod n = g^{xy} \bmod n$$

The value k is the secret key that both correspondents have exchanged and its size is 1024 bits (the size of the modulo n). The exponents x and y are often the same size as the prime n, but may be reduced to 160 and 256 bits. Thus a secret key has been negotiated on-line without transferring the key. Even by listening to all exchanges, it would be rather difficult to discover the key, unless there is a suitable way to calculate the discrete algorithm of X or of Y to rediscover the value of x or of y.

SSL uses the method called ephemeral Diffie–Hellman, where the exchange is short-lived, thereby achieving *perfect forward secrecy.* The Diffie–Hellman parameters are signed either with RSA or the DSA to guarantee integrity. The public keys of the various algorithms are included in the certificates that the certification authority has signed.

It should be noted that on March 29, 1997, the technique for key exchange entered the public domain.

5.13.2 Internet Security Association and Key Management Protocol

RFC 2408 [28] defines Internet Security Association and Key Management Protocol (ISAKMP), a generic framework to negotiate point-to-point security associations and to exchange key and authentication data between two parties. In ISAKMP, the term *security association* has two meanings. It is used to describe the secure channel established between two communicating entities. It can also be used to define a specific instance of the secure channel, i.e., the services, mechanisms, protocol, and protocol-specific set of parameters associated with the encryption algorithms, the authentication mechanisms, the key establishment and exchange protocols, and the network addresses. In ISAKMP, a domain of interpretation (DOI) is the context of operation in terms of the relevant syntax and semantics. RFC 2407 [26] defines the IP security DOI for security associations in IP networks within the ISAKMP framework.

ISAKMP specifies the formats of messages to be exchanged and their building blocks (payloads). A fixed header precedes a variable number of payloads chained together to form a message. This provides a uniform management layer for security at all layers of the ISO protocol stack, thereby reducing the amount of duplication within each security protocol. This centralization of the management of security associations has several advantages. It reduces connect setup time, improves reliability of software, and allows for future evolution when improved security mechanisms are developed, particularly if new attacks against current security associations are discovered.

To avoid subtle mistakes that can render a key exchange protocol vulnerable to attacks, ISAKMP includes five default exchange types. Each exchange specifies the content and the ordering of the messages during communications between the peers.

Although ISAKMP can run over TCP or UDP, many implementations use UDP on port 500. Because the transport with UDP is unreliable, reliability is built into ISAKMP.

The header includes, among other information, two 8-octet "cookies"—also called "syncookies"—which constitute an *anti-clogging* mechanism, because of their role against TCP SYN flooding. Each side generates a cookie specific to the two parties and assigns it to the remote peer entity. The cookie is constructed, for example, by hashing the IP source and destination addresses, the UDP source and destination ports and a locally generated secret random value. ISAKMP recommends including the data and the time in this secret value. The concatenation of the two cookies identifies the security association, and it gives some protection against replay of old packets or SYN flooding attacks. Protection against SYN flooding assumes that the attacker will not intercept the SYN/ACK

packets sent to the spoofed addresses used in the attack. As was explained earlier, the arrival of unsolicited SYN/ACK packets at a host that is accessible to the victim will elicit transmission of an RST packet, thereby telling the victim to free the allocated resources, so that the host whose address has been spoofed will respond by resetting the connection message [78, 85].

The negotiation in ISAKMP comprises two phases: (1) the establishment of a secure channel between the two communicating entities, and (2) the negotiation of security associations on the secure channel. For example, in the case of IPSec, Phase I negotiation is to define a key exchange protocol, such as the IKE and its attributes. Phase II negotiation concerns the actual cryptographic algorithms to achieve IPSec functionality.

IKE is an authenticated exchange of keys consistent with ISAKMP. IKE is a hybrid protocol that combines aspects of the Oakley Key Determination Protocol and of SKEME. Oakley utilizes the Diffie–Hellman key exchange mechanism with signed temporary keys to establish the session keys between the host machines and the network routers. SKEME is an authenticated key exchange that uses public key encryption for anonymity and nonrepudiation and provides means for quick refreshment [84]. IKE is the default key exchange protocol for IPSec

None of the data used for key generation is stored, and a key cannot be recovered after deletion, thereby achieving perfect forward secrecy. The price is a heavy cryptographic load, which becomes more important the shorter the duration of the exchanges. Therefore, to minimize the risks from denial of service attacks, ISAKMP postpones the computationally intensive steps until authentication is established.

Unfortunately, despite the complexity of IKE, the various documents that describe it do not use the best practices for protocol engineering. For example, there are no formal language descriptions, nor are there conformance test suites available. Nevertheless, IBM has revealed some details on the architecture of its implementation [87].

Although ISAKMP has been designed in a modular fashion, implementations are often not modular for commercial or legal reasons. For example, to satisfy the restrictions against the export of cryptographic software, Version 5.0 of Microsoft Windows NT had to sacrifice the modularity of the implementation. Similarly, the version that Cisco produces, which is based on the cryptographic library of Cylink Corporation, is only available in North America (United States and Canada). It should also be noted that the MIT distributes in North America the prototype of a version approved by the DOD. (*Note:* A new version of IKE is being prepared with the aim of removing problems that were uncovered. These problems relate to the hashing function and to the protection cookies.)

5.13.3 Simple Key Management for Internet Protocols

Simple Key Management for Internet Protocols (SKIP) is an approach to key exchange that Sun Microsystems championed at one time. The principle is to exchange a master key according to the Diffie–Hellman method, then store it in a cache memory to construct the encryption key for subsequent sessions. In this manner, the protocol avoids preliminary exchanges that are needed to define the secure channel before message exchange. This may be useful in applications where efficient use of the transmission bandwidth available justifies reduced security.

SKIP operates at the network layer. The IP packets that contain the information used in SKIP have an IP AH, and their payload is encapsulated according to the ESP procedures.

Although this method allows a reduction in the number of exchanges and alleviates the cryptographic loads, its success assumes that the master key is never compromised. Interest in SKIP seems to have subsided.

5.13.4 Key Exchange Algorithm

The Key Exchange Algorithm (KEA) is an algorithm from the U.S. National Security Agency (NSA). It is based on the Diffie–Hellman algorithm. All calculations in KEA are based on a prime modulus of 1024 bits generated as per the DSA specifications of FIPS 186.Thus, the key size is 1024 bits and, as in DSA, the size of the exponent is 160 bits.

KEA is used in the cryptographic PCMCIA card Fortezza and the SKIPJACK encryption algorithm. The experimental specifications of RFC 2773 [88] describe its use for securing file transfers with ftp. Those of RFC 2951 [89] provide security to telnet sessions.

Consider its use with telnet. The aim is to replace the user-level authentication through its log-in and password being exchanged in the clear with more secure measures and to be able to authenticate the server. It is known that a telnet session is a series of exchanges on a character-by-character basis. With the combination of KEA and SKIPJACK, the encryption of the telnet bit stream can be with or without integrity protection. Without the integrity service, each character corresponds to a single octet on-line. Stream integrity uses the one-way hash function SHA-1 and requires the transmission of 4 octets for every character, i.e., it adds an overhead of 300%. (*Note:* Version 2.0 of KEA is available from NIST at *http://csrc.nist.gov/encryption/skipjack-kea.htm.*)

5.14 CERTIFICATE MANAGEMENT

When a server receives a request signed with a public key algorithm, it must first authenticate the declared identity that is associated with the key. Next, it will verify if the authenticated entity is allowed to perform the requested action. Both verifications rely on one or more certificates that a certification authority has signed. These certificates can be used for identification and authentication, for privilege verification either on an identity basis or on the basis or an assigned role. As a consequence, certification and certificate management are the cornerstone of security in open networks.

The management of the infrastructure for certification can be decentralized or centralized. Decentralized certification utilizes Pretty Good Privacy (PGP) and is very popular among Internet users [90]. This model works by reference among users and, by obviating the need for a central authenticating authority, eliminates vulnerability to attacks on the central system and prevents the potential for power abuse, which are the weak points of centralized certification. Each user therefore determines the credence accorded to a public key and assigns the confidence level in the certificate that the owner of this public key has issued. Similarly, a user can recommend a new party to members of the same circle of trust. At one time, the World Wide Web Consortium (W3C) was favoring this approach in its Digital Signature Initiative. However, absence of any collective structure forces users to manage certificates by themselves (update, revocation, etc.). The load of this management increases exponentially with the number of participants, which makes this mode of operation impractical for large-scale operations.

Centralized certification is denoted X.509 certification, using the name of the ITU-T Recommendation [11] that defines the framework for authentication in open systems. X.509 is identical to ISO/IEC 9594-8, a joint standard from the ISO and the IEC. ANSI also ratified a corresponding standard known as ANSI X9.57 [91]. The focus of the following presentation will be on X.509 certificates. For details on certification practices, the interested reader is also invited to consult specialized books on certification, for example, Ford and Baum [92, pp. 357–404] whose first author was at the time a manager in VeriSign, one of the key players in certification.

The ITU-T and the ISO/IEC have established a whole series of recommendations to describe the operation of a public key infrastructure (PKI). These are:

- X.500 (ISO/IEC 9594-1) [66] for a general view of the concepts, the models, and the services.
- X.501 (ISO/IEC 9594-2) [93] for the different models used in the Directory.
- X.509 (ISO/IEC 9594-8) [11], which defines the framework for authentication through public key cryptography using identity certificates and attribute certificates.
- X.511 (ISO/IEC 9594-3) [94], which defines the abstract services of the Directory (search, creation, deletion, error messages, etc.).
- X.520 (ISO/IEC 9594-6) [95] and X.521 (ISO/IEC 9594-7) [96], which, respectively, specify selected attribute types (key words) and selected object classes to ensure compatibility among implementations.

These recommendations specify services, protocols, messages and object classes to carry out the following functions:

- Retrieval of credentials stored in the Directory by a directory user agent (DUA) at the client side and a directory system agent (DSA) at the server's side with the Directory Access Protocol (DAP) defined in X.519 (ISO/IEC 9594-5) [97].
- Distributed searches and referrals among directory system agents with the Directory System Protocol (DSP) of X.518 (ISO/IEC 9594-4) [98].
- Information sharing among directory system agents through replication of the directory using the Directory Information Shadowing Protocol (DISP) of X.525 (ISO/IEC 9594-9) [99].

The relationship among these different protocols is shown in Figure 5.13.

X.500 [66] is the basis for security directory services in the TMN, as defined in ANSI T1.252 [100], should such security be deemed necessary. T1.252 relies on the X.500 Directory for distribution of certified public keys to authorized entities.

In IP networks, a simplified version of DAP, the LDAP, is often used for communication between user agents and system agents. LDAP is the output of the Public Key Infrastructure (X.509) (PKIX) working group of the IETF and is defined in RFC2251 [101]. The main simplifications are as follows:

1. LDAP carried directly over the TCP/IP stack, thereby avoiding some of the OSI protocols at the application layer.

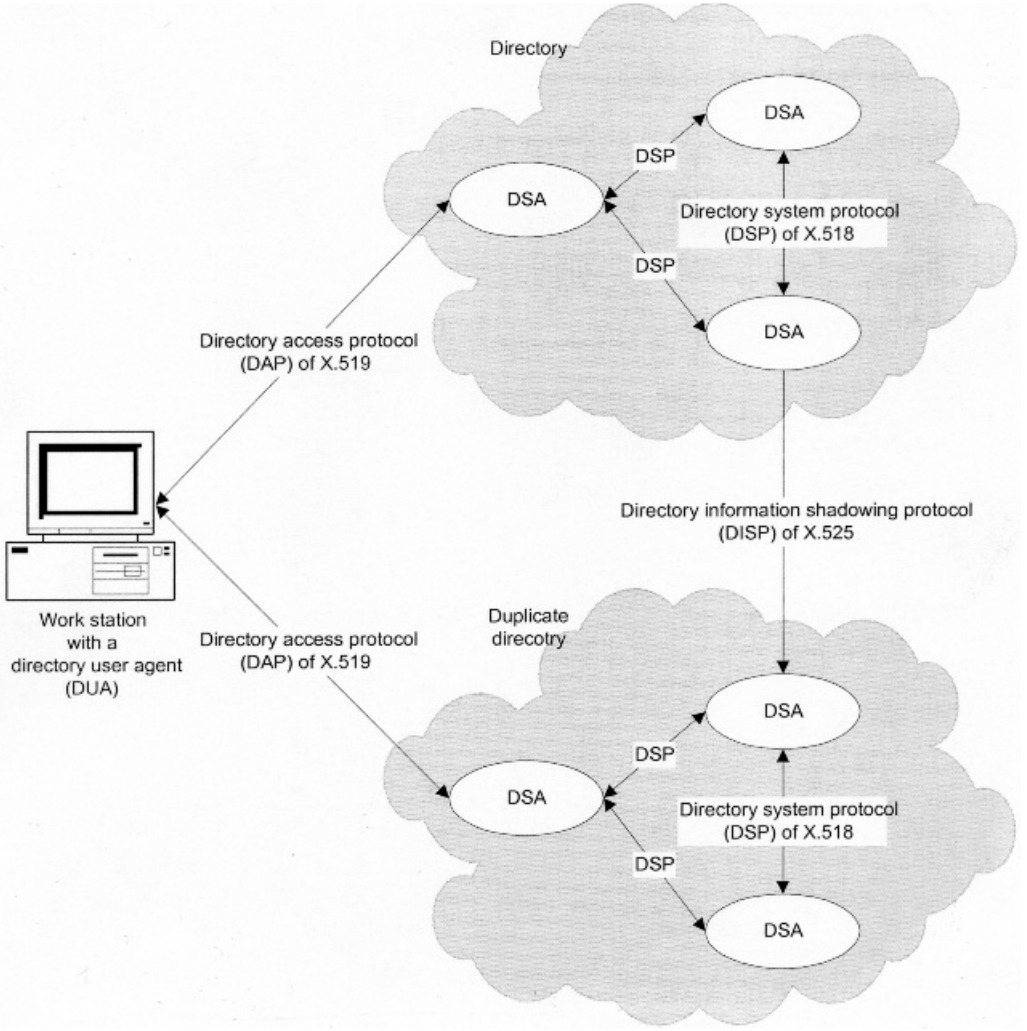

Figure 5.13 Communication protocols among the components of the directory system of X.500.

2. It uses simplified information models and object classes.
3. Being restricted to the client side, LDAP does not address what happens on the server side, for example, the duplication of the directory or the communication among servers.
4. Finally, Version 3 of LDAP, LDAPv3, does not mandate any strong authentication mechanism.

However, the latitude that LDAPv3 has allowed to developers with respect to strong authentication has resulted in some incompatibilities among different implementations of secure clients and servers. RFC 2829 [102] specifies a minimum subset of security functions common to all implementations of LSAPv3 that use the simple authentication and

security layer (SASL) mechanism defined in RFC 2222 [103]. SASL adds authentication services and, optionally, integrity and confidentiality. Simple authentication is based on the name/password pair, concatenated with a random number and/or a time stamp with integrity protection using MD5. Strong authentication is achieved on a session basis using the transport layer security (TLS) protocol.

5.14.1 Basic Operation

After receiving a request encrypted using public key cryptography, a server has to accomplish the following tasks before answering:

1. Reading of the certificate presented.
2. Verification of the signature by the certification authority.
3. Extraction of the requester public key from the certificate.
4. Verification of the requester signature on the request message.
5. Verification of the certificate validity by comparison with the certificate revocation lists (CRL).
6. Establishment of a certification path between the certification authority of the requester and the authority that the server recognizes.
7. Extraction of the name of the requester.
8. Determination of the privileges that the requester enjoys.

The certificate permits the accomplishment of tasks 1 through 7 of the preceding list. In the case of payments, the last step consists of verifying the financial data relating to the requester, in particular, whether the account mentioned has sufficient funds. In the general case, the problem is much more complex, especially if the set of possible queries is large. The most direct method is to assign a key to each privilege, which increases the difficulties of key management. This topic is currently the subject of intense investigation.

5.14.2 Description of an X.509 Certificate

An X.509 certificate [11] is a record of the information needed to verify the identity of an entity. This record includes the distinguished name of the user, which is a unique name that ties the certificate owner with its public key. The certificate contains additional fields to locate its owner's identity more precisely. Each version of X.509 [11] introduces its allotment of supplementary information, although compatibility with previous versions is retained. The essential pieces of information are those that can be found in the basic certificate (Version 1), whose content is illustrated in Table 5.5.

The certificate contains the digital signature using the private key of the certification authority. It is usually recommended that a distinct key be used for each security function (signature, identification, encryption, nonrepudiation, key encryption, key agreement, etc.). Accordingly, any given entity may have several certificates.

In the initial version of X.509 [11], the hierarchical arrangement of the distinguished names followed the rules for X.500 [66]. These rules where inspired by the worldwide assignment of telephone numbers in complete accordance with Recommendation X.400 for

Table 5.5 Content of the Basic X.509 Certificate

Field Name	Description
Version	Version of the X.509 certificate
serialNumber	Certificate serial number
Signature	Identifier of the algorithm used to sign the certificate and the parameters used
Issuer	Name of the certification authority
Validity	Duration of the validity of the certificate
Subject	User's references: distinguished name, unique identifier (optional), etc.
subjectPublicKeyInfo	Information concerning the public key algorithm of the sender, its parameters, and the public key itself

electronic mail. The directory entries are described using the key words defined in Recommendation X.520 [95], a partial list of which is given in Table 5.6.

The widespread use of the Internet has spawned other models for hierarchical naming. Version 3 of X.509, which was approved in 1996, has taken this fact into account and authorized the use of a variety of distinguished names, such as the network addresses, passport or identity card numbers, Social Security numbers, Internet domain names, email addresses, and uniform resource locators (URLs) for Web applications. The certificate can include additional pointers to the certified subject (physical name; postal address; electronic address) as well as identifiers related to specific applications, such as email address, EDI identity; or even personal details, such as profession, photo ID, and bank account number. This additional flexibility requires a name registration system to ensure that any name used unambiguously identifies a certificate subject. Without this verification automatic cross-checking of directory entries will be difficult, particularly on a worldwide basis.

Starting from Version 3 of X.509 (1996), the public key certificate can contain details on the security service for which the certified public key can be used, on the duration of its validity, on any restrictions on the use of the certificates, on cross-certifications with other certification authorities, etc. For example, X.509 now provides a way for a certificate issuer to indicate how the issuer's certificate policies can be considered equivalent to a different policy used by another certification authority (Section 8.2.2.7 of X.509 (2001) on policy mapping extension).

Version 4 of X.509 (2001) introduced several certificate extensions to improve the treatment of certificate revocation and to associate privileges with the identification public key certificates or with attribute certificates.

Table 5.6 Partial List of Key Words in X.520

Key Word	Meaning
C	Country
CN	Common name
L	Locality name
O	Organization name
OU	Organizational unit name

5.14.3 Certification Path

The idea behind X.509 [11] is to allow each user to retrieve the public key of certified correspondents so they can proceed with the necessary verifications. It is sufficient therefore to request the closest certification authority to send the public key of the communicating entity in a certificate sealed with the digital signature of that authority. This authority, in turn, relays the request to its own certifying authority, and this permits an escalation through the chain of authorities, or certification path, until reaching the top of the certification pyramid, where the root authority (RA) resides. Figure 5.14 depicts this recursive verification.

Armed with the public key of the destination entity, the sender can include a secret encrypted with the public key of the correspondent and corroborate that the partner is the one whose identity is declared. This is because, without the private key associated with the key used in the encryption, the destination will not be able to extract the secret. Obviously, for the two parties to authenticate themselves mutually, both users have to construct the certification path back to a common certification authority.

Thus, a certification path is formed by a continuous series of certification authorities between two users. This series is constructed with the help of the information contained in the directory by going back to a common point of confidence. The tree structure of the certification path can be hierarchical or nonhierarchical. (*Note:* As in the system for telephone numbering, each country or region can have its own local root authority. However, to ensure worldwide communication, agreements for cross-certification among the various authorities would extend the zone of validity of their certification, by making one certification authority the subject of a certificate from another authority.)

5.14.3.1 Hierarchical Certification Path According to the notational convention used in X.509 [11], a certificate is denoted by

$$\text{authority}\langle\langle\text{user}\rangle\rangle$$

Thus,

$$X_1\langle\langle X_2\rangle\rangle$$

indicates the certificate for user X_2 that authority X_1 has issued, while

$$X_1\langle\langle X_2\rangle\rangle\ X_2\langle\langle X_3\rangle\rangle \ldots X_n\langle\langle X_{n+1}\rangle\rangle$$

represents the certification path connecting user X_{n+1} to authority X_1. In other words, this notation is functionally equivalent to $X_1\langle\langle X_{n+1}\rangle\rangle$, which is the certificate that authority X_1 would have issued to user X_{n+1}. By constructing this path, another user would be able to retrieve the public key of user X_{n+1}, if that other user knows X_{1p}, the public key of authority X_1. This operation is called "unwrapping," and is represented by

$$X_{1p} \cdot X_1\langle\langle X_2\rangle\rangle$$

where $\cdot$ is an infix operator, whose left operand is the public key, X_{1p}, of authority X_1, and whose right operand is the certificate, $X_1\langle\langle X_2\rangle\rangle$, delivered to X_2 by that same certification authority. This result is the public key of user X_2.

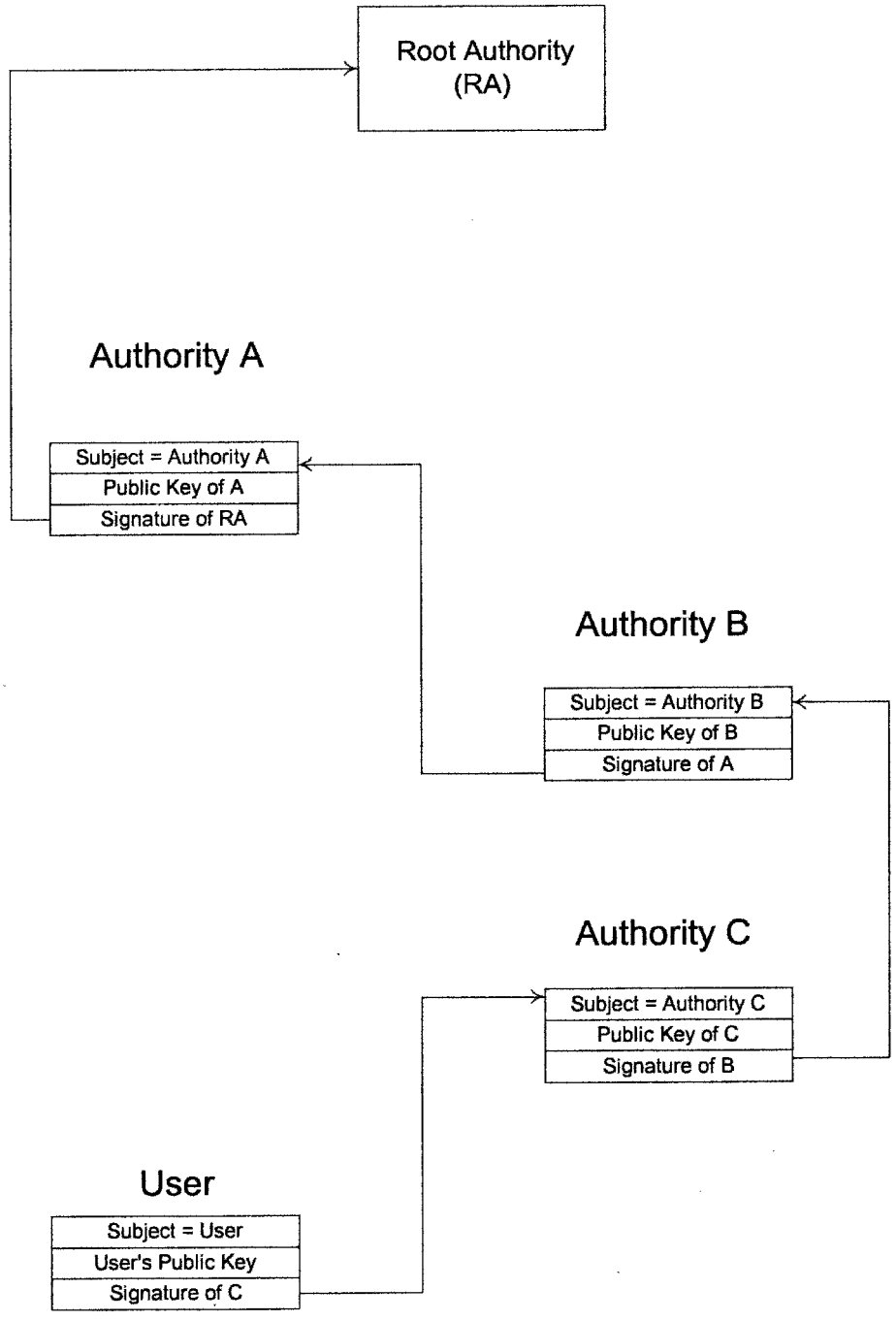

Figure 5.14 Recursive verification of certificates. (Adapted from Ford and Baum, *Secure Electronic Commerce,* Prentice Hall, 1997. With permission.)

In the example that Figure 5.15 depicts, assume that user A wants to construct the certification path toward another user, B. A can retrieve the public key of authority W with the certificate signed by X. At the same time, with the help of the certificate of V that W has issued, it is possible to extract the public key of V. In this manner, A would be able to obtain the chain of certificates:

$$X\langle\langle W\rangle\rangle, W\langle\langle V\rangle\rangle, V\langle\langle Y\rangle\rangle, Y\langle\langle Z\rangle\rangle, Z\langle\langle B\rangle\rangle$$

This itinerary, represented by $A \rightarrow B$, is the forward certification path that allows A to extract the public key Bp of B, by application of the operation · in the following manner:

$$Bp = Xp \cdot (A \rightarrow B) = Xp \cdot X\langle\langle W\rangle\rangle\, W\langle\langle V\rangle\rangle\, V\langle\langle Y\rangle\rangle\, Y\langle\langle Z\rangle\rangle\, Z\langle\langle B\rangle\rangle$$

In general, A also has to acquire the certificates for the return certification path $B \rightarrow A$, to send them to its partner:

$$Z\langle\langle Y\rangle\rangle, Y\langle\langle V\rangle\rangle, V\langle\langle W\rangle\rangle, W\langle\langle X\rangle\rangle, X\langle\langle A\rangle\rangle$$

When *B* receives these certificates from A, it can unwrap the certificates with its private key to extract the public key of A, Ap:

$$Ap = Zp \cdot (B \rightarrow A) = Zp \cdot Z\langle\langle Y\rangle\rangle\, Y\langle\langle V\rangle\rangle\, V\langle\langle W\rangle\rangle\, W\langle\langle X\rangle\rangle\, X\langle\langle A\rangle\rangle$$

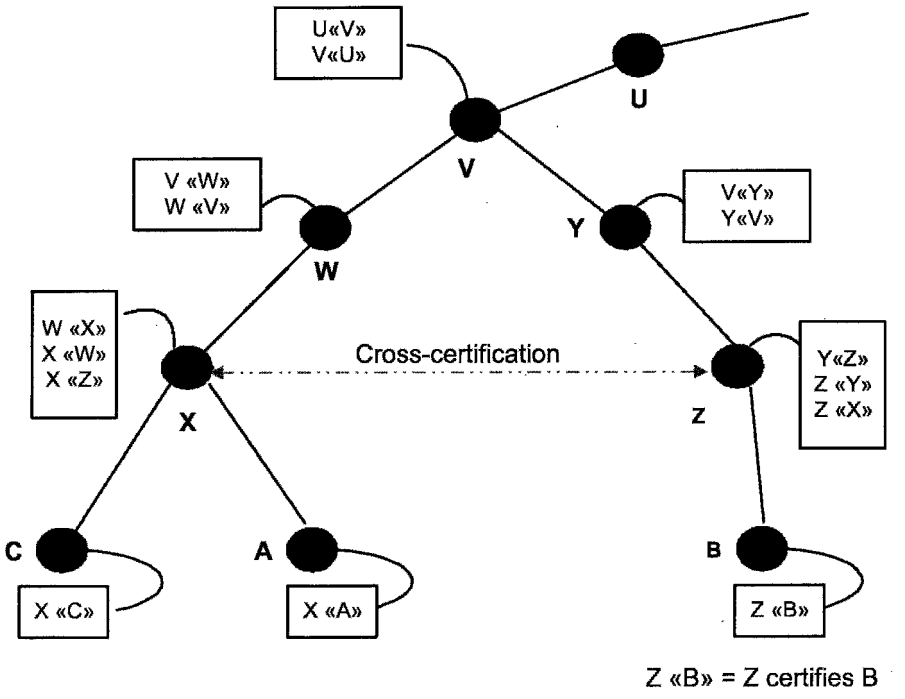

Figure 5.15 Hierarchical certification path according to X.509 [11]. (Adapted from ITU-T Recommendation X.509. Adapted from the International Telecommunication Union, 2000. With permission.)

As was previously mentioned, such a system does not necessarily impose a unique hierarchy worldwide. In the case of electronic payments, two banks or the fiscal authorities of two countries can mutually certify each other. In the preceding example, assume that authorities *X* and Z have cross-certified their respective certificates. If A wants to verify the authenticity of B, it is sufficient to obtain:

$$X\langle\langle Z\rangle\rangle, Z\langle\langle B\rangle\rangle$$

to form the forward certification path, and

$$Z\langle\langle X\rangle\rangle$$

to construct the reverse certification path. This permits the clients of the two banks to be satisfied with the certificates supplied by their respective banks.

5.14.3.2 Perspectives for Evolution As was previously mentioned, the X.509 directory [11] was inspired by the telephone directory. However, in the absence of a central authority, the equivalence for the Internet is not exact. To resolve these difficulties, the PKIX working group of the IETF has introduced the CMP of RFC 2510 and the OCSP of RFC 2560 to support X.509-based certification on the Internet. RFC 2585 [104] describes the conventions for using the File Transfer Protocol (ftp) and the HTTP to obtain certificates and certification revocation lists from their respositories.

Some authors question the validity of transposing the design of the telephone directory to this new environment. In their view, association of a telephone number and an identity is much more stable than the link between an entity and the public key certificates. Thus, attempts have been made to come up with simple distributed authentication structures such as the simple distributed security infrastructure (SDSI) that Ronald Rivest and Butler Lampson have proposed. The design of such a system revolves around keys and not entities; i.e., it is the access to a private key rather than the identity of an entity that plays the principal role for authentication.

Work in the simple public key infrastructure (SPKI) is focused on defining an authorization model rather than an authentication model. This model avoids the need for a global naming authority, as each local authority can accord authorization in its local domain without interference of other local authorities.

If certification authorities are not organized hierarchically, the users themselves would have to construct the certification path. In practice, the number of operations to be carried out can be reduced with various strategies, for example:

1. Two users served by the same certification authority have the same certification path, and the users can exchange their certificates directly. This is the case for entities *C* and A in Figure 5.15.
2. If one user is constantly in touch with users that a particular authority has certified, that user could store the forward and return certification paths in memory. This would reduce the effort for obtaining the other users' certificates to a query into the directory.
3. If two users know each other's certificates, they can mutually authenticate themselves without querying the directory. This reverse certification is based on the confidence that each user has in his or her own certification authority.

The interoperability of certificates in the general case is extremely complex, given the large number of potential certification authorities. However, by limiting the field of application, the task becomes less difficult. In the United States, NIST has published the Minimum Interoperability Specifications for PKI Components (MISPC) as NIST Special Publication 800-15 [105]. MISPC includes a certificate and a certificate revocation list profile, message formats, and basic transactions for a PKI issuing signature certificates. A reference implementation is also available. An updated specification is currently being developed.

In the same spirit and to facilitate electronic exchanges with suppliers of the federal agencies by replacing handwritten signatures as a means for authentication, the Canadian government has launched the program Government of Canada Public Key Infrastructure (GOCPKI). The objective of this program is to establish uniform criteria to manage the keys and certificates among all Canadian federal agencies.

In summary, the main difficulties in establishing a public key certification infrastructure that operates on a worldwide level are (1) the lack of harmonization among the authentication practices of the various certification authorities, (2) the absence of objective criteria to measure and evaluate the performance of the certification authorities, and (3) the absence of coordination among the multiple naming authorities.

5.14.4 Procedures for Strong Authentication

Having obtained the certification path and the other side's authenticated public key, X.509 [11] defines three procedures for authentication:

1. One-way or unidirectional authentication
2. Two-way or bidirectional authentication
3. Three-way or tridirectional authentication

5.14.4.1 One-Way Authentication One-way authentication takes place through the transfer of information from user A to user *B* according to the following steps:

- A generates a random number R^A used to detect replay attacks.
- A constructs an authentication token $M = (T^A, R^A, I_B, d)$ where T^A represents the time stamp of A (date and time) and I_B is the identity of B. T^A comprises two chronological indications, for example, the generation time of the token and its expiration date, and d is an arbitrary data. For additional security, the message can be encrypted with the public key of B.
- A sends B the message:

$$B \rightarrow A, A\{(T^A, R^A, I_B, d)\}$$

  where $B \rightarrow A$ is the certification path and $A\{M\}$ represents the message *M* encrypted with the private key of A.
- B carries on the following operations:
  —Obtain the public key of A, A_p, from $B \rightarrow A$, after verifying that the certificate of A has not expired.

—Recover the signature by decrypting the message A{M} with Ap;. *B* then verifies that this signature is identical to the message hash, thereby ascertaining simultaneously the signature and the integrity of the signed message.

—Verifies that *B* is the intended recipient.

—Verifies that the time stamp is "current."

—Optionally, verifies that R^A has not been previously used.

These exchanges prove:

- The authenticity of A, and that the authentication token has been generated by A.
- The authenticity of B, and that the authentication token has been intended for B.
- The integrity of the identification token.
- The originality of the identification token, i.e., that it has not been previously utilized.

5.14.4.2 Two-Way Authentication The procedure for two-way authentication adds to the previous unidirectionnal exchanges, similar exchanges but in the reverse direction. Thus:

- B generates another random number R^B.
- B constructs the message $M' = (T^B,R^B,I_A,R^A,d)$, where T^B represents the time-stamp of *B* (date and time), I_A is identity of A, and R^A is the random number received from A. T^B consists of one or two chronological indications, as previously described. For security, the message can be encrypted with the public key of A.
- B sends A the message:

$$B\{(T^B,R^B,I_A,R^A,d)\}$$

  where $B\{M'\}$ represents the message M' encrypted with the private key of B.
- A carries out the following operations:

  —Extracts the public key of *B* from the certification path and uses it to decrypt $B\{M'\}$ and recovers the signature of the message that *B* has produced. A verifies next that the signature is the same as the hashed message, thereby ascertaining the integrity of the signed information.

  —Verifies that A is the intended recipient.

  —Checks the time stamp to verify that the message is current.

  —As an option, verifies that RB has not been previously used.

5.14.4.3 Three-Way Authentication Protocols for three-way authentication introduce a third exchange from A to B. The advantage is the avoidance of time-stamping and, as a consequence, a trusted third party. The steps are the same as for two-way identification, but with $T^A = T^B = 0$. Then:

- A verifies that the value of the received R^A is the same as that sent to B.
- A sends *B* the message:

$$A\{R^B, I_B\}$$

encrypted with the private key of A.

- B performs the following operations:
  —Verifies the signature and the integrity of the received information.
  —Verifies that the received value of R^B is the same as that sent.

5.14.5 Certificate Revocation

Authentication establishes the correspondence between a public key and an identity only for a period of time. Therefore, certification authorities must refer to revocation lists, which contain certificates that have expired or have been revoked. These lists are continuously updated. Table 5.7 shows the format of the revocation list that Version 1 of X.509 has defined. The third revision of X.509 has added other optional entries such as the date of the certificate revocation and the reason for revocation.

In principle, each certification authority has to maintain at least two revocation lists: (1) a dated list of the certificates that it has issued and revoked, and (2) a dated list of all the certificates that the authorities know of and recognize as having been revoked. The root certification authority and each of its delegate authorities must be able to access these lists to verify the instantaneous state of all the certificates to be treated within the authentication system.

Revocation can be periodic or exceptional. When a certificate expires, the certification authority withdraws it from the directory (but retains a copy in a special directory, to be able to arbitrate any conflict that might arise in the future). Replacement certificates have to be ready and supplied to the owner to ensure the continuity of the service.

The root authority (or one of its delegated authorities) can cancel a certificate before its expiration date, for example, if the certificate owner's private key has been compromised or if there has been any abuse in usage. In the case of secure payments, the notion of solvency, i.e., that the user has available the necessary funds, is obviously one of the essential considerations.

The processing of the revocation lists must be speedy to alert users and, in certain countries, the authorities, particularly if the revocation is before the expiration date. Perfect synchronization among the various authorities must be attained to avoid questioning the validity of documents signed or encrypted before the withdrawal of the corresponding certificates.

Table 5.7 Basic Format of the X.509 Revocation List

Field	Comment
Signature	Identifier of the algorithm used to sign the certificates and the parameters used
Issuer	Name of the certification authority
thisUpdate	Date of the current update of the revocation list
nextUpdate	Date of the next update of the revocation list
revokedCertificates	References of the revoked certificates including the revocation date

Users must also be able to access the various revocation lists; this is not always possible because current client programs do not query these lists.

In summary, when an entity has a certificate signed by a certification authority, this means that the entry for that entity in the directory maintained by the certification authority has the following properties:

1. It establishes a relationship between the entity and a pair of public and private cryptographic keys.
2. It associates a unique distinguised name in the directory with the entity.
3. It establishes that, for a certain time, the authority is able to guarantee the correspondence between that unique distinguished name and the pair of keys.

5.14.6 Attribute Certificates

X.509 (Version 4) introduces a new type of public key certificates called *attribute certificates,* to link a subject to certain privileges separately from its authenticated identity. Attribute certificates allow the verification of the rights or prerogatives of their subject, such as access privileges [105]. Thus, once an identity has been authenticated with a public key certificate, the subject may use multiple attribute-certificates associated with that public key certificate.

Although it is quite possible to use public key identity certificates to define what the holder of the certificate may be entitled to, a separate attribute certificate may be useful in some cases, for example:

1. If the authority for privilege assignment is distinct from the certification authority.
2. A variety of authorities will be defining access privileges to the same subject.
3. The same subject may have different access permissions, depending on its role.
4. There is the possibility of delegation of privileges, in full or in part.
5. The duration of validity of the privilege is shorter than that of the public key certificate.

Conversely, the public key identity certificate may suffice for assigning privileges whenever:

1. The same physical entity combines the roles of certification authority and of attribute authority.
2. The expiration of the privileges coincides with that of the public key certificate.
3. Delegation of privileges is not permitted, or if permitted, all privileges are delegated at once.

The use of attribute certificates creates the need of a new infrastructure for their management. This is called privilege management infrastructure (PMI). When a single entity acts as both a certification authority and an attribute authority, it is strongly recommended that different keys be used for each kind of certificates.

The source of authority (SOA) is the trusted entity responsible for assigning access privileges. It plays a role similar to the root certification authority; however, the root certi-

fication authority may control the entities that can act as SOAs. An SOA can authorize the holder of a set of privileges to further delegate these privileges, in part or in full, along a *delegation path*. There may be restrictions on the power of delegation capability, for example, the length of the delegation path can be bonded and the scope of privileges allowed can be restricted downstream. To validate the delegation path, each attribute authority along the path must be checked to verify that it was duly authorized to delegate its privileges.

Attribute certification allows modification of the privileges of a role without impacts on the public key identity certificates However, privilege verification requires an independent verification of the privileges attributed to a role. This be done by prior agreement or through role-specification certificates. It is worth noting that hierarchical role-based access control allows role specifications to be more compact, because higher levels inherit the permissions accorded to subordinates.

X.509 [11] supports role-based access control (RBAC), provided that role specification certificates can be linked with the role assignments indicated in identity certificates or in attribute certificates. In addition, X.509 supports hierarchical RBAC through a "domination rule" that puts limits on the scope of delegated privileges. (*Note:* An X.509 RBAC policy for privilege management using XML is available at http://www.xml.org, and is based on work done at the University of Salford, U.K. [107].

5.15 APPLICATIONS FOR NETWORK MANAGEMENT

The first version of SNMP did not offer any security services. All management information can be accessed from the management system with read and write permissions. There are no mechanisms for authentication; messages are passed in plaintext form, and there are no sequence numbers. While the so-called "community string" could be used to identify the origin of the message, this string was not encrypted. SNMP packets are transmitted in the clear, which allows traffic analysis. Furthermore, there is no control for the integrity of the information; because there are no sequence numbers, replay attacks could not be fended off. Thus, the only way to protect Version 1 SNMP packets is through the use of IPSec.

Version 2 of SNMP allows confidentiality with DES as well as integrity verification through MD5. There is a mechanism for clock synchronization at both sides to prevent replay attacks. SNMPv2, however, does not provide a means for key distribution and management [108, pp. 285–286].

Version 3 of SNMP provides means for authentication, confidentiality, integrity verification, key management, access control, and clock synchronization. Authentication and integrity verification use the HMAC keyed-hashing algorithm, while confidentiality uses DES in the cipher block chaining (CBC) mode (see Appendix II).

A secret is constructed for communication between an SNMP manager and each of the SNMP agents that it manages . This secret is based on a nonshared secret stored in the SNMP manager and each agent's unique identifier using a hashing function (MD5 or SHA-1). The shared secret is then manually loaded in the SNMP agent. From this secret, the authentication key, the encryption key, and the initialization vector of the encryption can be derived [109, 110, pp. 202–203]. The specifications include a way for updating the various keys. SNMPv3 also offers a method for access control [111].

One major limitation in the security offered by SNMPv3 is that key update is based on the current key. Thus, if one key is compromised, an intruder can derive the next key by observing and decrypting the update exchanges. For small payloads, SNMPv3 requires about 24% more bandwidth than when SNMPv2c is secured with IPSec [112].

Finally, it has been observed in the past that many software implementations of SNMP managers do not decode the SNMP messages correctly or do not make the necessary syntax checks before interpreting and executing the commands. Thus, it is important to verify through adequate testing that the SNMP implementations behave correctly.

5.16 ENCRYPTION CRACKS

While the role of encryption is to mask the messages, the objective of cryptanalysis is to recover the message without knowledge of the encryption key. The basic approach consists of uncovering flaws in the cryptographic algorithms or in the system design that allows eavesdropping on the encrypted messages or at least spreading confusion.

The best-known cryptological attacks are of the following types:

1. Brute-force attacks where the assailant systematically tries all possible encryption keys until getting the one that will reveal the plain text.
2. Attacks on the encrypted text assuming that the clear text has a known given structure, for example, the systematic presence of a header with a known format (this is the case of email messages) or the repetition of known key words.
3. Attacks starting with the clear text, in total or in part, so as to uncover the encryption key.
4. Attacks starting with chosen plaintexts that are encrypted with the unknown key, so as to deduce the key itself.
5. Attacks by replaying old legitimate messages to evade the defense mechanisms and to short-circuit the encryption.
6. Attacks by interception of the messages (man-in-the-middle) where the interceptor inserts its eavesdrop at an intermediate point between the two parties. After interception, an exchange of a secret key, for example, the interceptor will be able to decipher the exchanged messages while the participants think they are communicating in complete security. The attacker may also be able to inject fake messages that would be treated as legitimate by the two parties.
7. Attacks by measuring the length of encryption times, of electromagnetic emissions, etc., to deduce the complexity of the operations, and hence their form.

Other techniques depend on the communication system itself. For example, corruption of the DNS can reorient packets to an attacker's address. Among the recommended measures to fend off attacks are the following [113]:

1. The explicit indication of the identity of the participants, if this identity is essential for the semantic interpretation of the message.
2. The choice of a sufficiently large key to discourage brute-force attacks, provided

that the encryption algorithm is well designed. The required key size grows with the computational power available to the adversaries.

3. The addition of random elements, a time stamp, and other nonce values that make replay attacks more difficult. However, deficient random-number generators open the possibility of attacks on secure algorithms.

In some cases, the physical protection of the whole cryptographic system (fibers, computers, smart cards, etc.) may be needed. For example, fiber bending results in a dispersion of 1–10% of the signal power; therefore, well-placed acoustic-optic devices can capture the diffraction pattern for later analysis.

In the real world, there are easier ways than cryptanalysis to break the cryptographic defenses. It is erroneous to evaluate the resistance of a cryptographic system by measuring the theoretical properties of the cryptographic algorithms used, without taking their practical implementation into account. Errors in design, gaps in implementations, or operational deficiencies, particularly if the encryption is done in software, augment the vulnerability of the system. It is well known, for example, that GSM, IEEE 802.11b, IS-41, etc., have faulty or deliberately weakened protection schemes. A catalog of the causes of vulnerability includes [113–115].

1. Nonverification of partial computations.
2. The use of defective random-number generators, because the keys and the session variables depend on a good supply source for nonpredictable bits.
3. The improper reutilization of random parameters.
4. The misuse of a hash function, which increases the chance of collisions.
5. The structural weakness of the telecommunications network.
6. The nonsystematic destruction of the clear text after encryption as well as the keys used in encryption.
7. The retention of the password or the keys in the virtual memory.
8. No checking of correct range of operation. This is particularly the case when buffer overflows can cause security flaws. Recently, a problem with Kerberos was discovered through buffer overflow within a process that administers the database.
9. The misuse of a protocol can lead to an authenticator traveling in plain text. For example, RFC 2109 [117] specifies that when the authenticator is stored in a cookie, the server has to set the Secure flag in the cookie header so that the client waits until a secure connection has been established with SSL or TLS before returning the cookie. Unfortunately, some Web servers neglect to set this flag, thereby negating that protection. The authenticator can also leak if the client software continues to use it even after the authentification is successful.

For example, when a program deletes a file, most commercial operating systems merely eliminate the corresponding entry in the index file. This allows recovery of the file, at least partially, with off-the-shelf software. The only means of guaranteeing total elimination of the data is to rewrite systematically each of the bits that the deleted file was using. Similarly, the use of the virtual memory in commercial systems exposes another vulnerability, because the secret document may be momentarily in the clear on the disk.

Systems for general commercial use must be easily accessible and affordably priced.

As a consequence, all the protective measures used in "top-secret" computers will not be used, and many compromises will be made to improve response time and the ease of use. However, if one starts from the principle that, sooner or later, any system is susceptible to unexpected attacks with unanticipated consequences, it would be useful to design the system such that any possible attack will be detected. For example, by accumulating proofs that are accepted by courts, the consequences would be alleviated and the possible damages reduced.

The starting point should be a correct definition of the type of expected threats and the eventual attack plans. The model has to take into account users' practices and the way they will be using the system, as well as the motivations for possible attacks. Such a realistic evaluation of threats and risks permits a precise understanding of what should be protected, against whom, and for how long.

5.17 SUMMARY

The task of securing telecommunications services has always been part of the role of network operators. There are two types of attacks: passive and active. Protection can be achieved with suitable mechanisms and appropriate policies. Recently, security has leaped to the forefront in priority because of changes in the regulatory environment and in technology. The fragmentation of operations that were once vertically integrated have increased the number of participants in end-to-end information transfer. In virtual private networks, customers are allowed some control of their part of the public infrastructure. Finally, security must be retrofitted in IP networks to protect from the inherent difficulties of having user traffic and network control traffic within the same pipe.

Security mechanisms can be implemented in one or more layers of the OSI mode. The choice of the layer depends on the security services to be offered and the coverage of protection.

Confidentiality guarantees that only the authorized parties can read the information transmitted. This is achieved by cryptography, whether symmetric or asymmetric. Symmetric cryptography is faster than asymmetric cryptography, but has a limitation in terms of the secure distribution of the shared secret. Asymmetric (or public key cryptography) overcomes this problem; this is why both can be combined. In on-line systems, public key cryptography is used for sending the shared secret that can be used later for symmetric encryption. Two of the public key schemes used for sharing the secrets are Diffie–Hellman and RSA. ISAKMP is a generic framework to negotiate point-to-point security and to exchange key and authentication data among two parties.

Data integrity is the service for preventing nonauthorized changes to the message content during transmission. A one-way hash function is used to produce a signature of the message that can be verified to ascertain integrity. Blind signature is a special procedure for signing a message without revealing its content.

Identification of participants depends on whether cryptography is symmetric or asymmetric. In asymmetric schemes, there is a need for authentication using certificates. In the case of human users, biometric features can be used for identification in specific situations. Kerberos is an example of a distributed system for on-line identification and authentication using symmetric cryptography.

Access control is used to counter the threats of unauthorized operations. There are

two types of access control mechanisms: identity-based and role-based. Both can be managed through certificates defined by ITU-T Recommendation X.509 [11]. Denial of service is the consequence of failure of access control. These attacks are inherently associated with IP networks where network control data and user data share the same physical and logical bandwidths. The best solution is to authenticate all communications by means of trusted certificates. Short of this, defense mechanisms will be specific to the problem at hand.

Nonrepudiation is a service that prevents a person who has accomplished an act from denying it later. This is a legal concept that is defined through legislation. The service comprises the generation of evidence, their recording, and subsequent verification. The technical means to ensure nonrepudiation include electronic signature of documents, the intervention of third parties as witnesses, time-stamping, and sequence numbering of the transactions.

APPENDIX I: AREAS RELATED TO SECURITY POLICIES

Security policies cover several areas such as [16, 38]:

- Policies regarding physical security of sites and network components, including the threats from fires, earthquakes, floodings, etc, and responses to emergencies.
- Policies for *prevention,* including physical access security, personnel risk analysis, security screening, access to management information.
- Policies for *administering* cryptographic keys for network elements or certificates, etc.
- Policies for *intrusion detection:* through usage pattern analysis to detect theft of service or denial of service attacks, network security alarm software intrusion audit.
- Policies for *audits*: what should be in an event record (nature of event, time, etc.), who can specify them, how to analyzed audit trails, etc.
- Policies for *reports,* i.e., the capability of reporting events to the network management system in real time as selected by a network administrator.
- Policies for *containment.*
- Policies for *recovery,* for example, backup policies.

APPENDIX II: PRINCIPLES OF SYMMETRIC ENCRYPTION

AII.1 Modes of Algorithm Utilization for Block Encryption

The four modes for using symmetric algorithms of the block cipher type are (1) ECB mode; (2) CBC mode, (3) cipher feedback (CFB) mode, and (4) output feedback (OFB) mode.

The ECB mode is the most obvious, because each clear block is encrypted independently of the other blocks. However, this mode is susceptible to attacks by replay of

blocks, which results in the perturbation of the messages even without breaking the code. This is the reason this mode is only used to encrypt random data, such as the encryption of keys during authentication.

The other three modes have in common that they protect against such types of attacks with a feedback loop. They also have the additional property that they need an initialization vector to start the computations. These values can be revealed. The difference between the three feedback modes resides in the way the clear text is mixed, partially or in its entirety, with the preceding encrypted block.

In the CBC mode, the input to the encryption module is the clear text mixed with the preceding encrypted block with an exclusive OR. This encryption operation is represented in Figure AII.1, and Figure AII.2 represents the decryption. In these figures, M_i represents the ith block of the clear message, while E_i is the corresponding encrypted block. Thus, the encrypted block, E_i, is given by

$$E_i = E_k(M_i \oplus E_{i-1}), \qquad i = 0, 1, \ldots$$

where $E_K(\ )$ represents the encryption with the secret key, K, and $\oplus$ is the exclusive OR operation. The starting value E_o is the initialization vector The decryption operation, shown in Figure AII.2 is described by

$$M_i = E_{i-1} \oplus D_K(E_i)$$

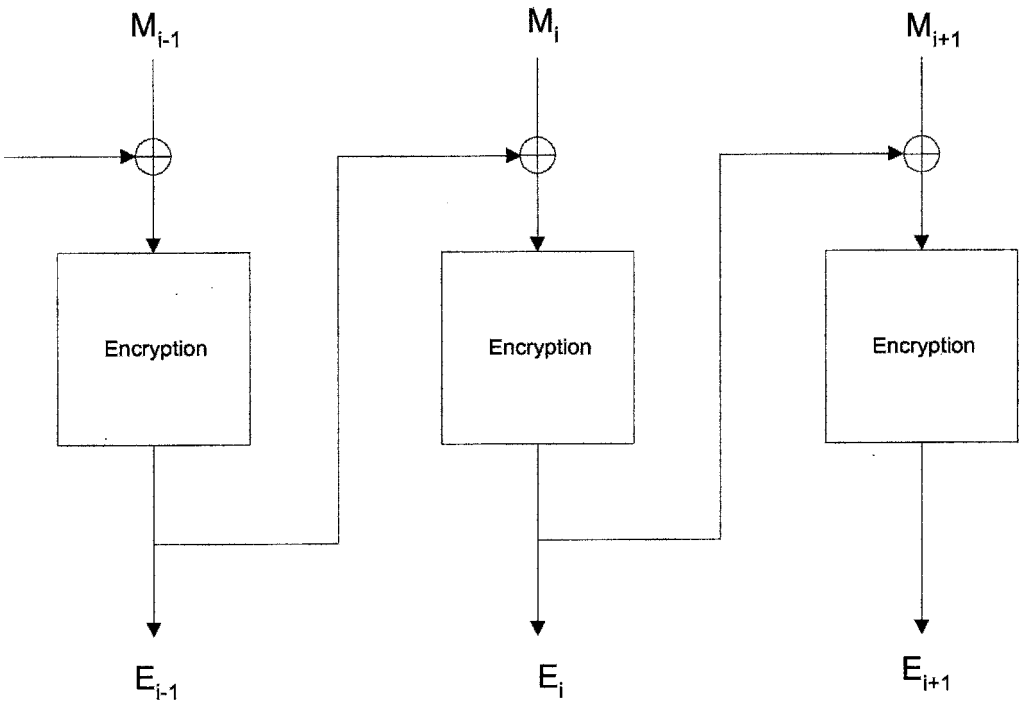

Figure AII.1 Encryption in the CBC mode.

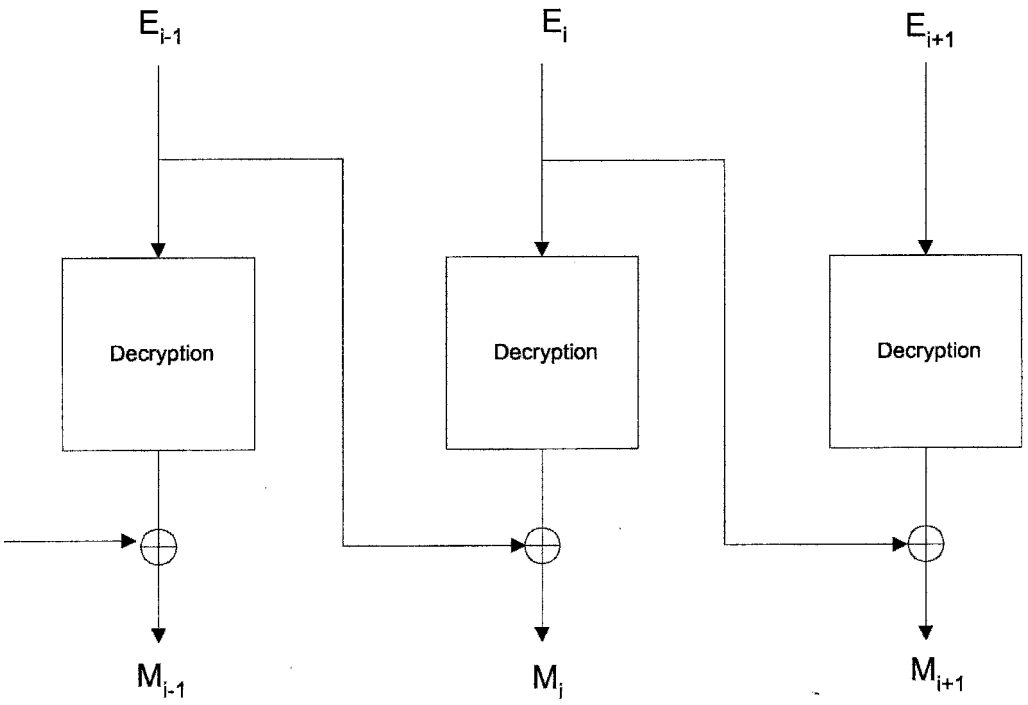

Figure AII.2 Decryption in the CBC mode.

The CBC mode is generally useful for non-real-time encryption of files, for example, to calculate the signature of a message (or its MAC). In fact, this is the method indicated in the various standards for securing financial and banking transactions: ANSI X9.9 [118], ANSI X9.19 [119], ISO 8731-1 [120], and ISO/IEC 9797 [121], as well as in the ESP protocol of IPSec

The CFB and OFB modes are more appropriate for the real-time encryption of a character stream, such as in the case of a client connected to a server.

In CFB encryption, the encryption of a block of clear text of m bits is done in units of n bits (n = 1, 8, 32, or 64 bits), with $n \leq m$, in n/m cycles. At each cycle, n bits of the clear message, M_i, are combined, with the help of an exclusive OR, with the leftmost n bits of the previously encrypted block, E_{i-1}, to yield the new n bits of the new encrypted block E_i. These same n bits are then concatenated to the feedback bits in a shift register, and then all the bits of this register are shifted n positions of the left. The n leftmost bits of the register are ignored, while the remainder of the register content is encrypted, and the n leftmost bits are used in the encryption of the next n bits of the clear text. The decryption operation is identical with the roles of M_i and E_i transposed. Figure AII.3 depicts the encryption, and Figure AII.4 illustrates the decryption.

It can be seen that the block encryption algorithm is acting on both sides. The decryption operation is sensitive to bit errors, because one bit error in the encrypted text affects the decryption of (m/n + 1) blocks, the present one and the next (m/n). In this mode of operation, the initialization vector needs to be changed after each message to prevent cryptanalysis.

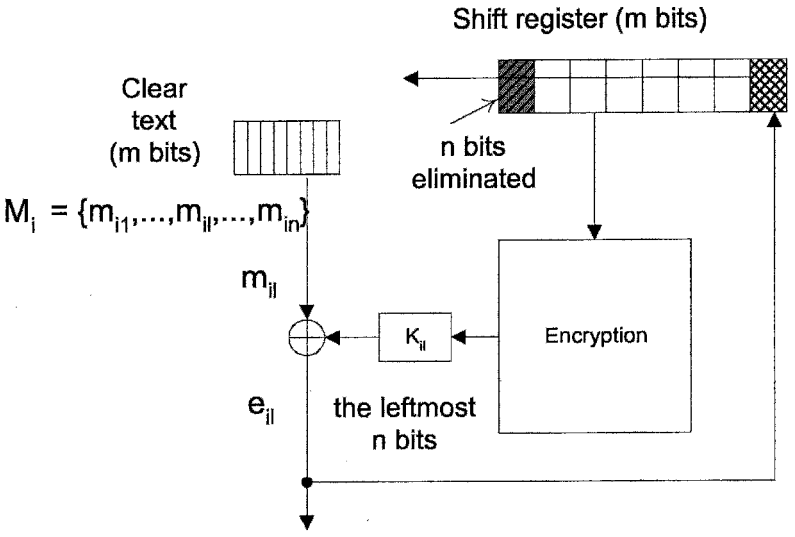

Figure AII.3 Encryption in the CFB mode of a block of m bits and n bits of feedback.

In the case $n = m$, the shift register can be eliminated and the encryption is done as illustrated in Figure AII.5. Thus, the encrypted block, E_i, is given by

$$E_i = M_i \oplus E_K(E_{i-1})$$

where $E_K(\ )$ represents the encryption with the secret key K.

The decryption is obtained with another exclusive OR operation as follows:

$$M_i = E_i \oplus E_K(E_{i-1})$$

which is shown in Figure AII.6.

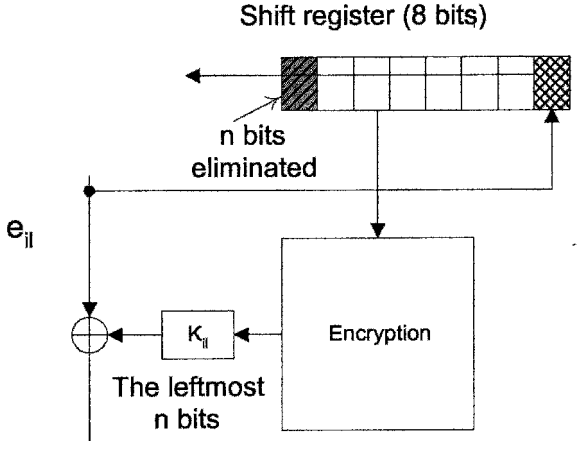

Figure AII.4 Decryption in the CFB mode of a block of m bits with n bits in the feedback loop.

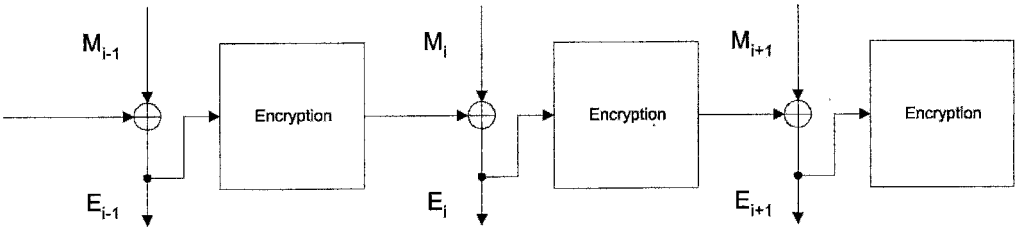

Figure AII.5 Encryption in the CFB mode for a block of n bits with a feedback of n bits.

The CFB mode can be used to calculate the MAC of a message as the last block encrypted two consecutive times. This method is also indicated in ANSI X9.9 [116] for the authentication of banking messages, as well as ANSI X9.19 [117],, ISO 8731-1 [118], and ISO/IEC 9797 [119]. In the encryption of a telnet stream with SKIPJACK $m = 64$ bits and $n = 32$ or 8 bits, depending on whether integrity is provided. These modes are denoted as CFB-8 without integrity and CFB-32 with integrity.

Finally, the OFB mode is similar to the CFB mode, except that the n bits in the feedback loop result from the encryption and are not in the ciphertext transmitted to the destination. This is illustrated in Figures AII.7 and AII.8 for the encryption and decryption, respectively.

OFB is adapted to situations where the transmission systems insert significant errors, because the effects of such errors are confined: a single bit error in the ciphertext affects only one bit in the recovered text. However, to avoid the loss of synchronization, the values in the shift registers should be identical. Thus, any system that incorporates the OFB mode must be able detect the loss of synchronization and have a mechanism to reinitialize the shift registers on both sides with the same value.

The encryption operation is represented in Figure AII.9, for the case where $n = m$, and is described by

$$E_i = M_i \oplus E_i$$

$$S_i = E_K(S_{i-1})$$

The algorithm approaches a permutation of m bits that, on average, repeats itself every $2^m - 1$ cycles. Therefore, it is recommended to utilize mode OFB only with $n = m$, i.e., the feedback size equal to the block size, to increase the security of the operation.

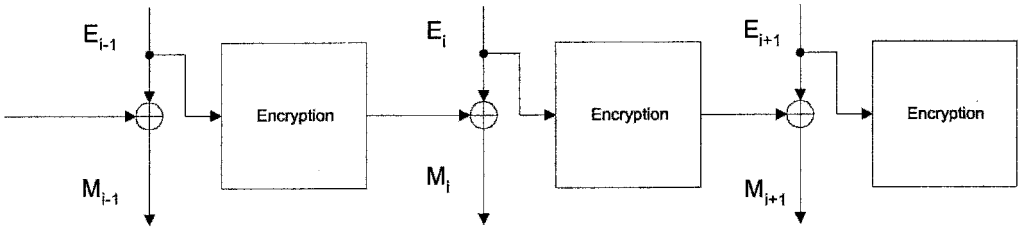

Figure AII.6 Decryption in the CFB mode for a block of n bits with a feedback of n bits.

The decryption is described by

$$M_i = E_i \oplus S_i$$

$$S_i = E_K(S_{i-1})$$

and it takes place as indicated in Figure AII.10.

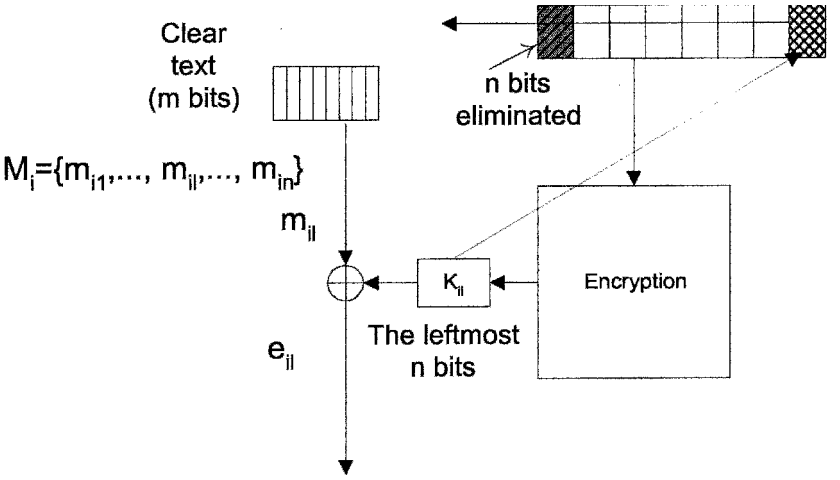

Figure AII.7 Encryption in OFB mode of a block of *m* bits with a feedback of *n* bits.

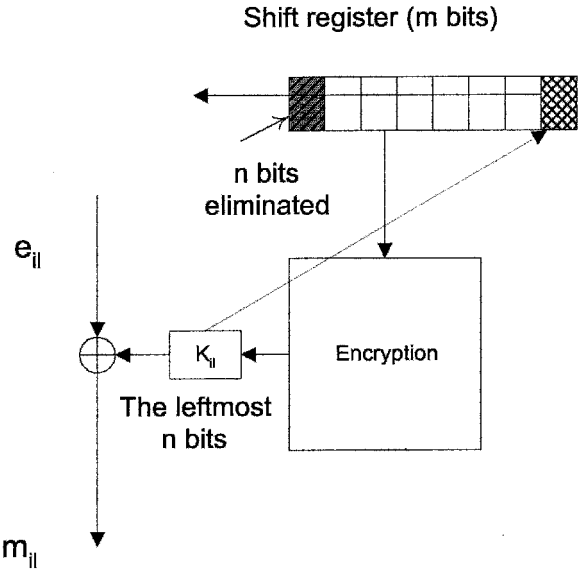

Figure AII.8 Decryption in OFB mode of a block of *m* bits with a feedback of *n* bits.

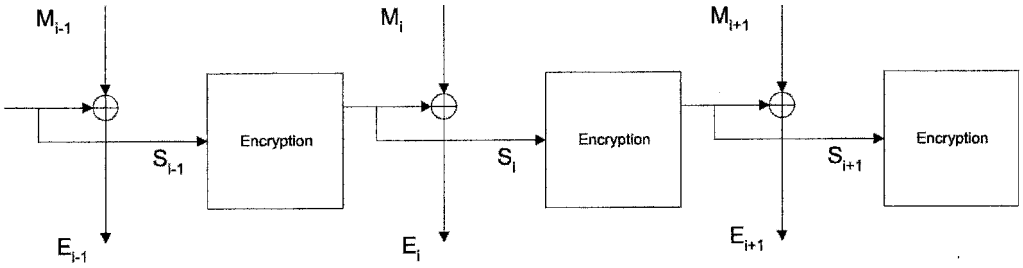

Figure AII.9 Encryption in OFB mode with a block of *n* bits and a feedback of *n* bits.

AII.2 Examples of Symmetric Block Encryption Algorithms

AII.2.1 Advanced Encryption Standard The AES is the new symmetric encryption algorithm that will replace DES. It is published by NIST as FIPS 197 and is based on the algorithm Rijndael that was developed by two Belgian cryptographers. It is a block code with blocks of 128, 192, or 256 bits. The corresponding key lengths are128, 192, and 256 bits, respectively.

The selection in October 2000 came after two rounds of testing following NIST's invitation to cryptographers from around the world to submit algorithms. In the first round, 15 algorithms were retained for evaluation. The submissions came from a variety of companies, such as Deutsche Telekom, IBM, NTT, RSADSI, from Canada and South Korea, as well as from independent researchers. All the algorithms in competition operated with key lengths of 128, 192, and 256 bits. In the second round of evaluation, five finalists were retained : RC6, MARS, Rijnadel, Serpent, and Twofish. Results from the evaluation and the rational for the selection have been documented in a public report by NIST [120].

AII.2.2 Data Encryption Standard DES is one of the most widely used algorithms in the commercial world for applications such as the encryption of financial documents, the management of cryptographic keys, and the authentication of electronic transactions. This algorithm was developed by IBM and then adopted as a U.S. standard in 1977. It was published in FIPS 81, then adopted by ANSI in ANSI X3.92 [40] under the name of *Data Encryption Algorithm*. This algorithm has reached the end of its useful life and is expected to be replaced by the AES.

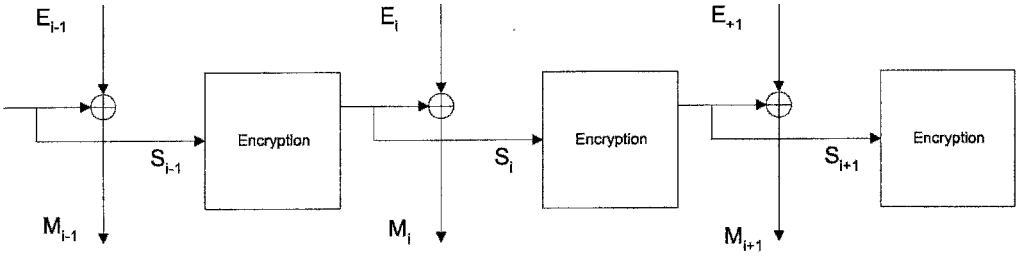

Figure AII.10 Decryption in OFB mode for a block of *n* bits with a feedback of *n* bits.

DES operates by encrypting blocs of 64 bits of clear text to produce blocks of 64 bits of ciphertext. The encryption and decryption are based on the same algorithm with some minor differences in the generation of subkeys.

The key length is 64 bits, with 8 bits for parity control, which gives an effective length of 56 bits. The operation of DES consists of 16 rounds of identical operations, each round including a text substitution followed by a bit-by-bit permutation of the text, based on the key. If the number of rounds is fewer than 16, DES can be broken by a clear text attack, which is easier to conduct than an exhaustive search.

AII.2.3 Triple DES The vulnerability of DES to an exhaustive attack has encouraged the search of other, surer algorithms until a new standard is available. Given the considerable investment in the software and hardware implementations of DES, triple DES uses DES three successive times with two different keys. Figure AII.11 represents the schema used in triple DES.

The use of three stages doubles the effective length of the key to 112 bits. The operations "encryption–decryption–encryption" aim at preserving compatibility with DES, because if the same key is used in all operations, the first two cancel each other. As there are several ways to attack the algorithm, it is recommended that three independent keys be used [39, pp. 359–360].

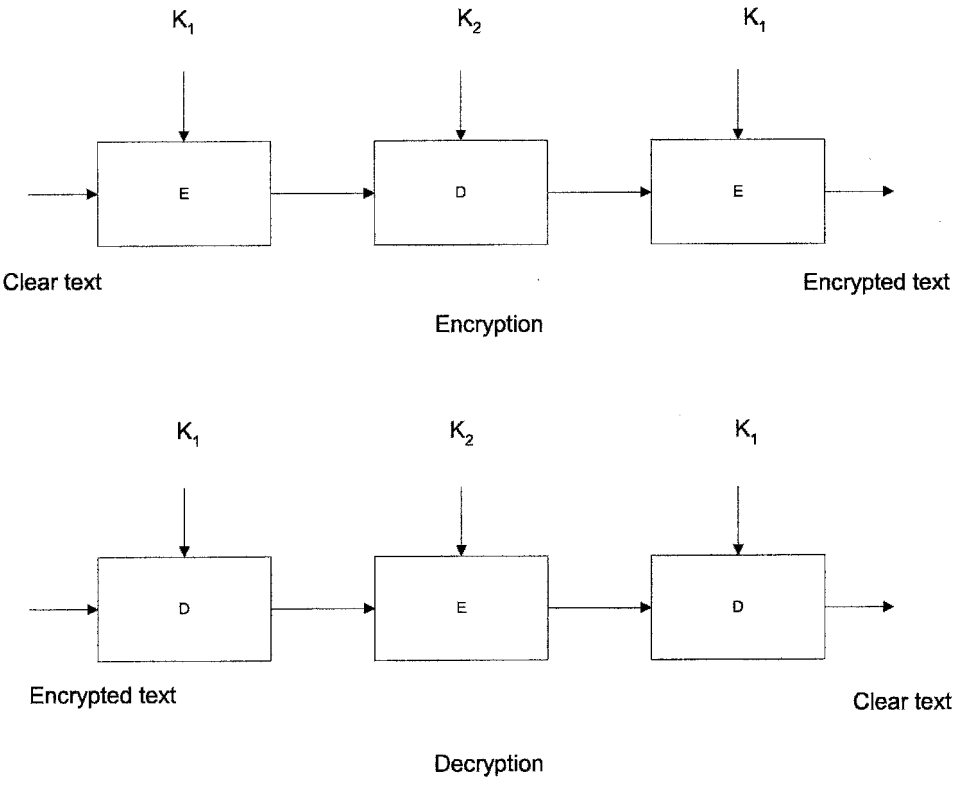

Figure AII.11 Operation of triple DES.

AII.2.4 International Data Encryption Algorithm IDEA was invented by Xuejia Lai and James Massey circa 1991 [45]. The algorithm takes blocks of 64 bits of the clear text, divides them into subblocks of 16 bits each, and encrypts them with a key 128 bits long. The same algorithm is used for encryption and decryption. IDEA is clearly superior to DES, but has not been a commercial success. The patent is held by a Swiss company, Ascom-Tech AG, and is not subject to U.S. export control.

AII.2.5 SKIPJACK SKIPJACK is an algorithm developed by NSA for several single-chip processors, such as Clipper, Capstone, and Fortezza. Clipper is a tamper-resistant very large-scale integrated (VLSI) chip used to encrypt voice conversation. Capstone provides the cryptographic functions needed for secure electronic commerce, and is used in Fortezza applications. SKIPJACK is an iterative block cipher with a block size of 64 bits and a key of 80 bits. It can be used in any of the four modes ECB, CBC, CFB (with a feedback of 8, 16, 32, or 64 bits), and OFB with a feedback of 64 bits.

APPENDIX III: PRINCIPLES OF PUBLIC KEY ENCRYPTION

The most popular algorithms for public cryptography are those of Rivest, Shamir, and Adleman (RSA) [50], Rabin (1979), and ElGamal [57]. Nevertheless, the overwhelming majority of proposed systems in commercial systems are based on the RSA algorithm.

It should be noted that RSADSI was founded in 1982 to commercialize the RSA algorithm for public key cryptography. However, its exclusive rights ended with the expiration of the patent on September 20, 2000.

AIII.1 RSA

Consider two odd prime numbers **p** and **q** whose product $\mathbf{N} = \mathbf{p} \times \mathbf{q}$. The values **p** and **q** are kept secret, while **N** is the modulus used in the computation which is public. When referring to the key size for RSA, what is meant is the length of the modulus **N** in bits.

Let $\varphi(\mathbf{n})$ be the Euler totient function of **N**. By definition, $\varphi(\mathbf{n})$ is the number of elements formed by the complete set of residues that are relatively prime to **N**. This set is called the reduced set of residues modulo **N**.

If **N** is a prime, $\varphi(\mathbf{N}) = \mathbf{N} - 1$. However, because $\mathbf{N} = \mathbf{p} \times \mathbf{q}$ by construction, while **p** and **q** are primes, then

$$\varphi(\mathbf{N}) = (\mathbf{p} - 1)\,(\mathbf{q} - 1)$$

According to Fermat's little theorem, if m is a prime, and **a** is not a multiple of **m** (for example, $\mathbf{a} < \mathbf{m}$), the

$$\mathbf{a}^{\mathbf{m-1}} \equiv 1 \pmod{m}$$

Euler generalized this theorem in the form:

$$\mathbf{a}^{\varphi(\mathbf{N})} \equiv 1 \pmod{\mathbf{N}}$$

Choose the integers **e**, **d** both less that $\varphi(\mathbf{N})$ such that the greatest common divisor of $(\mathbf{e}, \varphi(\mathbf{N})) = 1$ and $\mathbf{e} \times \mathbf{d} \equiv 1 \bmod (\varphi(\mathbf{N})) = 1 \bmod ((\mathbf{p} - 1)(\mathbf{q} - 1))$.

Let **X**, **Y** be two numbers less than **N**,

$$\mathbf{Y} = \mathbf{X}^{\mathbf{e}} \bmod \mathbf{N} \qquad \text{with} \qquad 0 \equiv \mathbf{X} < \mathbf{N}$$

$$\mathbf{X} = \mathbf{Y}^{\mathbf{d}} \bmod \mathbf{N} \qquad \text{with} \qquad 0 \equiv \mathbf{Y} < \mathbf{N}$$

because, by applying Fermat's little theorem,

$$\mathbf{Y}^{\mathbf{d}} \bmod \mathbf{N} = (\mathbf{X}^{\mathbf{e}})^{\mathbf{d}} \bmod \mathbf{N} = \mathbf{X}^{\mathbf{ed}} \bmod \mathbf{N} = \mathbf{X}\varphi^{(\mathbf{N})} \equiv 1 \ (\text{mod } \mathbf{N}) = 1 \bmod \mathbf{N}$$

To start the process, a block of data is interpreted as an integer. To do so, the total block is considered as an ordered sequence of bits (of length, say, λ). The integer is considered to be the sum of the bits by giving the first bit the weight of $2^{\lambda-1}$, the second bit the weight of $2^{\lambda-2}$, and so on until the last bit, which will have the weight of $2^0 = 1$.

The block size must be such that the largest number does not exceed the modulo **N**. Incomplete blocks must be completed by padding bits with either 1 or 0 bits. Further padding blocks may be also added.

The public key of the algorithm **Pk** is the number **e**, along with **N**, while the secret key **Sk** is the number **d**. RSA achieves its security from the difficulty of factoring **N**. The number of bits of **N** are considered to be the key size of the RSA algorithm. The selection of the primes **p** and **q** must make this factorization as difficult as possible.

Once the keys have been generated, it is recommended, for reasons of security, that the values of **p** and **q** as well as all intermediate values, such as the product $(\mathbf{p} - 1)(\mathbf{q} - 1)$, be deleted. Nevertheless, the preservation of the values of **p** and **q** locally can double or even quadruple the speed of decryption.

AIII.1.1 Practical Considerations To increase the speed of signature verification, suggested values for the exponent **e** of the public key are 3 or $2^{16} + 1$ (65,537) [40, p. 437]. Other variants designed to speed up decryption and signing are discussed in Boneh and Shacham [122].

For short-term confidentiality, the modulus **N** should be at least 768 bits. For long-term confidentiality (5 to 10 years), at least 1024 bits should be used. Currently, it is believed that confidentiality with a key of 2048 bits would last about 15 years.

AIII.2 Public Key Cryptography Standards

Public Key Cryptography Standards (PKCS) are business standards developed by RSA Laboratories in collaboration with many other companies working in the area of cryptography. They have been used in many aspects of public key cryptography that are based on the RSA algorithm. At the time of writing this section, their number has reached 15.

PKCS #1 (RFC 2437 [58]) defines the mechanisms for data encryption and signature using the RSA algorithm. These procedures are then utilized for constructing the signatures and electronic envelopes described in PKSC #7. In particular, PKCS #1 defines an encryption scheme based on the optimal asymmetric encryption padding (OAEP) of Bellare and Rogaway [124]. PKCS #2 and #4 have been incorporated in PKCS #1.

PKCS #3 defines the key exchange protocol using the Diffie–Hellman algorithm.

PKCS #5 describes a method for encrypting an information using a secret key derived from a password. For hashing, the method utilizes either MD2 or MD5 to compute

the key starting with the password, and then encrypts the key with DES in the CBC mode.

PKCS #6 is a syntax for X.509 certificates.

PKCS #7 (RFC 2315 [125]) defines the syntax of a message encrypted using the Basic Encoding Rules (BER) of ASN.1 [126] of ITU-T Recommendation X.209 [127]. These messages are formed with the help of six content types:

1. *Data,* for clear data
2. *SignedData,* for signed data
3. *EnvelopedData,* for clear data with numeric envelopes
4. *SignedAndEnvelopedData,* for data that are signed and enveloped
5. *DigestedData,* for digests
6. *EncryptedData,* for encrypted data

The secure messaging protocol, Secure Multipurpose Internet Mail Extensions (S/MIME), as well as the messages of the SET protocol, designed to secure bankcard payments over the Internet utilize the PKSC #7 specifications.

PKCS #8 describes a format for sending information related to private keys.

PKCS #9 defines the optional attributes that could be added to other protocols of the series. The following items are considered: the certificates of PKCS #6, the electronically signed messages of PKCS #7, and the information on private keys as defined in PKCS #8.

PKCS #10 (RFC 2896 [128]) describes the syntax for certification requests to a certification authority. The certification request must contain details on the identity of the candidate for certification, the distinguished name of the candidate, his or her public key, and optionally, a list of supplementary attributes, a signature of the preceding information to verify the public key, and an identifier of the algorithm used for the signature so that the authority could proceed with the necessary verifications. The version adopted by the IETF is called Cryptographic Message Syntax (CMS).

PKCS #11 defines a cryptographic interface, called *Cryptoki* (Cryptographic Token Interface Standard), between portable devices such as smart cards or PCMCIA cards and the security layers.

PKCS #12 describes a syntax for the storage and transport of public keys, certificates, and other user's secrets. Microsoft utilizes this syntax in the new version of NT Server 5.0.

PKCS #13 describes a cryptographic system using elliptic curves.

PKCS #15 describes a format to allow the portability of cryptographic credentials such as keys, certificates, passwords, PINs, among application and among portable devices such as smart cards. (*Notes:* (1) Even though the specifications of PKCS #1, #7, and 10 have been described in IETF documents, this organization has not accepted them as standards because they mandate the utilization of algorithms that RSADSI does not offer free of charge. (2) In PKCS #11 and #15, the word *token* is used to indicate a *portable device capable of storing persistent data.*)

AIII.3 Pretty Good Privacy

PGP is considered to be the commercial system whose security is closest to military grade. It is described in one of the IETF documents, namely, RFC 1991 [129]. PGP consists of six functions:

1. A public key exchange using RSA with MD5 hashing.
2. A data compression with ZIP, which reduces the file size and redundancies before encryption. Reduction of the size augments the speed for both processing and transmission, while reduction of the redundancies makes cryptanalysis more difficult.
3. Message encryption with IDEA.
4. Encryption of the user's secret key using the digest of a sentence instead of a password.
5. An ASCII "armor" is used to protect the binary message for any mutilations that might be caused by Internet messaging systems. This armor is constructed by dividing the bits of three consecutive octets into four groups of 6 bits each and then by coding each group using a 7-bit character according to a given table. A checksum is then added to detect potential errors;
6. Message segmentation.

Although the IETF has worked on PGP, it has not adopted PGP as a standard yet because it incorporates protocols that have patent protections, such as IDEA and RSA. Current activities in the IETF attempt to use the framework of PGP, but with protocols that circumvent these restrictions.

AIII.4 Elliptic Curve Cryptography

Elliptic curves have been studied in algebraic geometry and number theory. They have been applied in factoring integers, in primality proving, in coding theory, and in cryptography [130]. Elliptic curve cryptography (ECC) is a public key cryptosystem where the computations take place on an elliptic curve. These cryptosystems are variants of the Diffie–Hellman and DSA algorithms, thereby giving rise to the Elliptic Curve Diffie–Hellman algorithm (ECDH) and the Elliptic Curve Digital Signal Algorithm (ECDSA), respectively. They can be used to create digital signatures and to establish keys for symmetric cryptography. The ECDSA algorithm is now an ANSI standard (X9.62) [131].

The elliptic curves are defined over the finite field of the integers modulo a primary number p (the Gallois field GF(p)) or that of binary polynomials ($GF(2^m)$). The key size is the size of the prime number or the binary polynomial in bits. Cryptosystems over $GF(2^m)$ appear to be slower than over GF(p), but there is no consensus on that point. Their main advantage, however, is that additions over $GF(2^m)$ do not require integer multiplications, which reduces the cost of the integrated circuits implementing the computations.

ECDSA is used for digital signing, while ECDH can be used to secure on-line key exchange. Perfect forward secrecy is achieved with the ephemeral mode of ECDH, i.e., the key is short-term. Diffie–Hellman and ECDH are comparable in speed, but RSA is much slower because of the generation of the key pair.

Typical key sizes are in the range 160 to 200 bits. The advantage of elliptic curve cryptography is that key lengths are shorter than for existing public key schemes that provide equivalent security. For example, the level of security of 1024-bit RSA can be achieved will elliptic curves with a key size in the range of 171–180 bits [132]. This is an important factor in wireless communications and whenever bandwidth is a scarce resource.

Table AIII.1 gives various computations times for digital signatures with RSA, DSA, and ECDSA on a 200-MHz Pentium Pro [133]. The results show that RSA is slower for signing and much faster for signature verification than DSA and ECDSA. Thus, from a

Table AIII.1 Computation Times for Digital Signatures with the RSA, DSA, and ECDSA Algorithms [133]

	Timings in ms (on a 200-MHz Pentium Pro)		
Operation	RSA with $N = 1024$ and $e = 3$	DSA with 1024 bits	ECDSA over GF(p) with 168 bits
Sign	43	7	5
Verify	0.6	27	19
Key generation	1100	7	17
Parameter generation	0	6500	High

Table AIII.2 Comparison of Public Key Systems in Terms of Key Length in Bits for the Same Security Level [130]

RSA	Elliptic Curve	Reduction Factor RSA/ECC
512	106	5:1
1024	160	7: 1
2048	211	10:1
5120	320	16:1
21000	600	35:1

computational speed viewpoint, RSA is more suitable for certificate verification, while Diffie–Hellman, ECDH, and ECDSA are more suitable for on-line communication.

Finally, Table AIII.2 compares the key lengths of RSA and elliptic cryptography for the same amount of security measured in terms of effort to break the system [130].

APPENDIX IV: PRINCIPLES OF THE DIGITAL SIGNATURE ALGORITHM

According to the DSA defined in ANSI X9.30:1 [1], the signature of a message M is the pair of numbers r and s computed as follows:

$$r = (g^k \bmod p) \bmod q$$

and

$$s = \{k^{-1}[H(\mathrm{M}) + xr]\} \bmod q,$$

where

- p and q are primes such that $2^{511} < \mathrm{p} < 2^{1024}$, $2^{159} < q < 2^{160}$, and q is a prime divisor of $(p - 1)$, i.e., $(p - 1) = mq$ for some integer m.
- $g = h^{(p-1)/q} \bmod p$ is a generator polynomial modulo p of order q, with h any integer $1 < h < (p - 1)$ such that $h^{(p-1)/q} \bmod p > 1$. By Fermat's little theorem, $g^q = h^{(p-1)/q} \bmod p = 1$, since $g < p$. Thus, each time the exponent is a multiple of q, the result will be equal to 1 (mod p).

- x and k are randomly generated integers between 0 and q (i.e., $0 < x, k < q$).
- x is the private key of the sender, while the public key y is given by $y = g^x \bmod p$.
- k^{-1} is the multiplicative inverse of $k \bmod q$, i.e., $(k^{-1} \times k) \bmod q = 1$, where $0 < k, k^{-1} < q$.
- $H(\ )$ is the SHA-1 hash function.

To verify the signature the verifier computes

$$w = s^{-1} \bmod q$$

$$u_1 = H(\mathrm{M})\, w \bmod q$$

$$u_2 = rw \bmod q$$

$$v = (g^{u1}\, y^{u2} \bmod p) \bmod q$$

If $v = r$, the signature is valid.

To show this we have:

$$\begin{aligned} v &= \{(g^{[H(M)w \bmod q]}\, y^{rw \bmod q}) \bmod p\} \bmod q \\ &= (g^{[H(M)w \bmod q]}\, g^{xrw \bmod q}) \bmod p\} \bmod q \\ &= \{g^{[H(M)+xr]w \bmod q] \bmod p}\} \bmod q \\ &= (g^{ksw \bmod q} \bmod p) \bmod q \\ &= (g^{k \bmod q} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q, \text{ since the generator is of order } q \text{ by construction,} \\ &= r \end{aligned}$$

Note that the random variable k is also transmitted with the signature. This means that if the verifier knows the signer's private key, they will be able to pass additional information through the channel established through the value of k.

REFERENCES

1. ANSI X9.30:1, "American National Standard—Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA)" (Revision of X9.30:1-1995), American Bankers Association, 1997.
2. ITU, *Management Framework for Open Systems Interconnection for CCITT Applications,* Recommendation X.700, 1992.
3. ITU, *Security Architecture for Open Systems Interconnection for CCITT Applications,* Recommendation X.800, 1991.
4. S. Covaci, L. Marchisio, and D. J. Milham, "Trouble Ticketing *X* Interfaces for International Private Leased Data Circuits and International Freephone Service," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS'98),* vol. 2, pp. 342–353, February 1998.
5. D. J. Milham, C. Hatch, A. Hensen, S.-T. Johnsen, and R. Moons, "European ATM Service Introduction—OSS Interconnection Between Operators," in *Proceedings of IEEE Network Operations and Management Symposium (NOMS 2000),* pp. 247–260, 2000.

6. ITU, *Architecture for Customer Network Management Service for Public Data Networks,* Recommendation X.160, 1996.
7. ITU, *Definition of Customer Network Management Service for Public Data Networks,* Recommendation X.161, 1997.
8. ITU, *Definition of Management Information for Customer Network Management Service for Public Data Networks to be Used with the CNMc Interface,* Recommendation X.162, 1996.
9. M. H. Sherif and S. Ho, "Evolution of Operation Support Systems in Public Data Networks," in *Proceedings of the 5th IEEE Symposium on Computers and Communications ISCC2000,* Antibes-Juan Les Pins, France, pp. 72–77, July 2000.
10. ISO/IEC TR 13335-5, *Information Technology—Guidelines for the Management of IT Security—Part 5: Management Guidance on Network Security,* 2001.
11. ITU, *Information Technology—Open Systems Interconnection—The Directory: Public-key and Attribute Certificate Frameworks,* Recommendation X.509 (ISO/IEC 9594-8), 2000.
12. ISO 7498-2, *Information Technology*—Open Systems Interconnection—Basic Reference Model—Part 2: Security Architecture, 1989.
13. R. W. Baldwin and, C. V. Chang, "Locking the e-Safe," *IEEE Spectrum,* vol. 34, no. 2, pp. 40–46, 1997.
14. C. E. Perkins, *Mobile IP: Design Principles and Practices,* Addison-Wesley, Reading, Massachusetts, 1998.
15. W. Ford and B. O'Higgins, "Public-key Cryptography and Open Systems Interconnection," *IEEE Communications Magazine,* vol. 39, no. 7, pp. 30–35, 1992.
16. S. E. Forrester, M. J. Palmer, D. C. McGlaughlin, and M. J. Robinson, "Security in Data Networks," *BT Technol. J.,* vol. 16, no. 1, pp. 52–75, 1998.
17. P. Rolin, "La sécurité dans les réseaux," in *Réseaux de Communication et Conception de Protocoles,* G. Juanole, A. Sehrouchni, and D. Seret, Eds., Hermès, Paris, pp. 80–103, 1995.
18. A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL Protocol Version 3.0," available at http://www.netscape.com/PROD/eng/ssl3/ssl-toc.html, 1996.
19. T. Dierks and C. Allen, "The TLS Protocol Version 1.0," IETF RFC 2246, January 1999.
20. T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of Power Analysis Attacks on Smartcard," in *Proceedings of USENIX Workshop on Smartcard Technology,* Chicago, pp. 151–161, May 1999.
21. W. Simpson, "The Point-to-Point Protocol (PPP)," IETF RFC 1661, July 1994.
22. W. Townsley, A. Valencia, A Rubens, G. Pall, G. Zorn, and B. Palter, "Layer Two Tunneling Protocol L2TP," IETF RFC 2661, August 1999.
23. S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," IETF RFC 2401, November 1998.
24. S. Kent and R. Atkinson, "IP Authentication Header," IETF RFC 2402, November 1998.
25. S. Kent and R. Atkinson, "IP Encapsulating Security Payload (ESP)," IETF RFC 2406, November 1998.
26. D. Piper, "The Internet IP Security Domain of Interpretation for ISAKMP," IETF RFC 2407, November 1998.
27. D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)," IETF RFC 2408, November 1998.
28. D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," IETF RFC 2409, November 1998.
29. P. Glenn and S. Kent, "The NULL Encryption Algorithm and Its Use with IPSEC," IETF RFC 2410, November 1998.

30. R. Thayer, N. Doraswamy, R. Glenn, "IP Security Document Roadmap," IETF RFC 2411, November 1998.
31. H. Orman, "The OAKLEY Key Determination Protocol," IETF RFC 2412, November 1998.
32. C. Huitema, *IPv6: The New Internet Protocol,* Prentice Hall, Englewood Cliffs, New Jersey, 1996.
33. N. Doraswamy and D. Harkins, *IPSec: The New Security Standard for the Internet, Intranets, and Virtual Private Networks,* Prentice Hall, Upper Saddle River, New Jersey, 1999.
34. B. Patel, B. Adoba, W. Dixon, G. Zorn, and S. Booth, "Securing L2TP using IPsec," IETF RFC 3193, November 2001.
35. A. Carasik, "Secure Shell FAQ," Revision 1.4, http://www.tigerlair.com/ssh/faq, February 2001.
36. T. Ylönen, "The SSH (Secure Shell) Remote login Protocol," http://www.tigerlair.com/ssh/faq/ssh1-draft.txt, November 1995.
37. T. Ylönen, "SSH- Secure login Connections over the Internet," in *Proceedings of the Sixth USENIX Security Symposium,* pp. 37–42, 1996.
38. ATM Forum Technical Committee, "Methods for Securely Managing ATM Network Elements—Implementation Agreement," AF-SEC-0179.000, Version 1.0, April 2002.
39. B. Schneier, *Applied Cryptography,* John Wiley & Sons, New York, second edition, 1996.
40. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone *Handbook of Applied Cryptography,* CRC Press, Boca Raton, Florida, 1997.
41. ANSI X3.92, "American National Standard—Data Encryption Algorithm (DEA)," 1981.
42. ANSI X3.105, "American National Standard—Data link encryption," 1983.
43. ANSI X3.106, "American National Standard—Data Encryption Algorithm, Modes of Operations," 1983.
44. ISO 8372, *Information processing—Modes of Operation for a 64-bit Block Cipher Algorithm,* 1987.
45. X. Lai, J. Massey, and S. Murphy, "Markov Ciphers and Differential Cryptanalysis," in *Proceedings of Eurocrypt'91,* LNCS 547, Springer-Verlag, Berlin, pp. 17–38, 1991.
46. X. Lai and J. Massey, "A Proposal for a New Block Encryption Standard," in *Proceedings of Eurocrypt'90,* LNCS 473, Springer-Verlag, Berlin, pp. 389–404, 1991.
47. R. L. Rivest, "The RCE Encryption Algorithm," *Crypto Bytes,* vol. 1, no. 1, pp. 9–11. 1995.
48. ANSI X9.52, "American National Standard—Triple Data Encryption Algorithm Modes of Operation," 1998.
49. W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory,* vol. 22, no. 6, pp. 644–654, 1976.
50. R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM,* vol. 21, no. 2, pp. 120–126, 1978.
51. R. Rivest, "The MD4 Message Digest Algorithm," IETF RFC 1320, April 1992.
52. R. Rivest, "The MD5 Message Digest Algorithm," IETF RFC 1321, April 1992.
53. ISO/IEC 10118-3, *Information Technology—Security Techniques—Hash-Functions—Part 3: Dedicated Hash-Functions,* 1998.
54. H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A Strengthened Vervion of RIPMD," in *Proceedings of Fast Software Encryption Workshops,* LNCS 1039, Springer-Verlag, Berlin, pp. 71–82, 1996.
55. ANSI X9.30:2, "American National Standard—Public Key Cryptography for the Financial Services Industry: Part 2: The Secure Hash Algorithm 1 (SHA-1)," (Revised), American Bankers Association, 1993.
56. P. van Oorschot and M. Wiener, "Parallel Collision Search with Applications to Hash Func-

tions and Discrete Logarithms," in *Proceedings of 2nd ACM Conference on Computer and Communications Security,* pp. 210–218, November 1994.

57. T. ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology—CRYPTO'84 Proceedings,* Springer-Verlag, Berlin, pp. 10–18, 1985.
58. B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0," IETF RFC 2437, October 1998.
59. P. Chaum, "Blind Signatures for Untraceable Payments," in *Crypto82,* Plenum Press, New York, pp. 199–203, 1983.
60. P. Chaum, "Privacy Protected Payments: Unconditional Payer and/or Payee Untraceability," in *SmartCard 2000,* D. Chaum and J. Schaumüler-Bichl, Eds., Elsevier Science Publishers B.V. (North Holland), Amsterdam, pp. 69–93, 1989.
61. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Advances in Cryptology—CRYPTO'96 Proceedings,* Springer-Verlag, Berlin, pp. 1–15, 1996.
62. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication," IETF RFC 2104, February 1997.
63. S. A. Sherman, R. Skibom, and R. S. Murray, "Secure Network Access Using Multiple Applications of AT&T's Smart Card," *AT&T Technical Journal,* pp. 61–72, September/October 1994.
64. S. Nanavati, M. Thieme, and R. Nanavati, *Biometrics: Identity Verification in a Network World,* John Wiley & Sons, New York, 2002.
65. ANSI X9.84 "American National Standard—Biometric Information Management and Security," 2001.
66. ITU, *Information Technology—Open Systems Interconnection—The Directory: Overview of Concepts, Models and Services,* Recommendation X.500 (ISO/IEC 9594-1), 2001.
67. ITU, *Information Technology—Open Systems Interconnection—Security Architecture for Open Systems: Authentication Framework,* Recommendation X.811 (ISO/IEC 10181-3), 1995.
68. ITU, *Information Technology—Open Systems Interconnection—Security Architecture for Open Systems: Access Control Framework,* Recommendation X.812 (ISO/IEC 10181-3), 1995.
69. C. Rigney, S. Willens, A. Rubens, and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)," IETF RFC 2865, June 2000.
70. C. Finseth, "An Access Control Protocol, Sometimes Called TACACS," IETF RFC 1492, July 1993.
71. J. Hill, "An Analysis of the RADIUS Authentication Protocol," http://www.untruth.org/~josh/security/radisu/radisu-auth.html, 2001.
72. R. K. C. Chang, "Defending Against Flooding-based Distributed Denial-of-Service Attack: A Tutorial," *IEEE Communications Magazine,* vol. 40, no. 10, pp. 42–51, 2002.
73. D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," in *Proceedings of the 10th USENIX Security Symposium,* Washington, D.C., August 2001.
74. M. H. Sherif, G. R. Ash, and J. Han, "Transparent Processing of Resource Management (RM) Cells that Are Not Defined in ATM-TM-011–21.00, I.371, or I.361," ATM Forum Contribution 00–479, Phoenix, Arizona, January 2001.
75. C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a Denial of Service Attack on TCP," in *Proceedings of IEEE Symposium on Security and Privacy,* pp. 208–223, 1997.
76. D. E. Comer, *Interworking with TCP/IP Vol I: Principles, Protocols, and Architecture,* Prentice Hall, Englewood Cliffs, New Jersey, third edition, 1995.
77. *CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow in IIS Indexing*

Service DLL, CERT/CC CA-2001-19, last revised January 17, 2002; available at http://www.cert.org/advisories/CA-2001-19.html.

78. A. Juels, and J. Brainard, "A Cryptographic Countermeasure Against Connection Depletion Attacks," in *Proceedings of the Internet Society Symposium on Network and Distributed System Security,* S. Kent, Ed., IEEE Computer Society Press, New York, pp. 151–165, 1999 (also available on www.rsasecurity.com).
79. ITU, *Security Architecture for Open Systems: Non-Repudiation Framework,* Recommendation X.813 (ISO/IEC 10181-3), 1995.
80. W. Fumer and P. Landrock, "Principles of Key Management," *IEEE Journal on Selected Areas in Communications,* vol. 11, no. 5, pp. 785–793, 1993.
81. B. C. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications. Magazine,* vol. 32, no. 9, pp. 33–38, 1994.
82. M. A. Sirbu and J. C.-I. Chuang, "Distributed Authentication in Kerberos Using Public Key Cryptography," 1996; available at http://www.cs.cmu.edut/afs/andrew.cmu.edu/inst/ini.
83. M. Sirbu and J. Chuang, "Public-Key Based Ticket Granting Service in Kerberos," IETF RFC 1510, May, 1996.
84. B. Tung, *Kerberos: A Network Authentication System,* Addison Wesley Longman, Reading, Massachusetts, 1999.
85. W. A. Simpson, "IKE/ISAMP Considered Harmful," *;login:* vol. 24, no. 6, 48–58, 1999.
86. H. Krawczyk, "SKEME: A Versatile Secure key Exchange Mechanism for Internet," in *Proceedings of the Internet Society Symposium on Network and Distributed System Security,* IEEE Computer Society Press, New York, pp. 114–127, 1996.
87. P.-C. Cheng, "An Architecture for the Internet Key Exchange Protocol," *IBM Systems Journal,* vol. 40, no. 3, pp. 721–746, 2001.
88. R. Housley, P. Yee, and W. Nace, "Encryption Using KEA and SKIPJACK," IETF RFC 2773, February 2000.
89. R. Housley, T. Horting and P. Yee, "TELNET Authentication Using KEA and SKIPJAK," IETF RFC 2951, September 2000.
90. S. L. Garfinkel, *PGP: Pretty Good Privacy,* O'Reilly and Associates, Sebastopol, California, 1995.
91. ANSI X9.57, "American National Standard—Public Key Cryptography for the Financial Services Industry: Certificate Management" (Revision of X9.57-1995), 1997.
92. W. Ford and M. S. Baum, *Secure Electronic Commerce: Building the Infrastructure,* Prentice Hall, Englewood Cliffs, New Jersey, 1997.
93. ITU, *Information Technology—Open Systems Interconnection—The Directory: Models,* Recommendation X.501 (ISO/IEC 9594-2), 2001.
94. ITU, *Information Technology—Open Systems Interconnection—The Directory: Overview of Concepts, Models and Services,* Recommendation X.511 (ISO/IEC 9594-3), 2001.
95. ITU, *Information Technology—Open Systems Interconnection—The Directory: Selected Attribute Types,* Recommendation X.520 (ISO/IEC 9594-6), 2001.
96. ITU, *Information Technology—Open Systems Interconnection—The Directory: Selected Object Classes,* Recommendation X.521 (ISO/IEC 9594-7), 2001.
97. ITU, *Information Technology—Open Systems Interconnection—The Directory: Protocol Specifications,* Recommendation X.519 (ISO/IEC 9594-5), 2001.
98. ITU, *Information Technology—Open Systems Interconnection—The Directory: Procedures for Distributed Operation,* Recommendation X.518 (ISO/IEC 9594-4), 2001.
99. ITU, *Information Technology—Open Systems Interconnection—The Directory: Replication,* Recommendation X.525 (ISO/IEC 9594-9), 2001.

100. ANSI T1.252, "Operations, Administration, Maintenance and Provisioning (OAM&P)—Security for the Telecommunications Management Network Directory," 1996.
101. M. Whal, T. Howes, and S. Kille, "Lightweight Directory Access Protocol (v3)," IETF RFC 2251, December 1997.
102. M. Wahl, H. Alvestrand, J. Hodges, and R. Morgan, "Authentication Methods for LDAP," IETF RFC 2829, May 2000.
103. J. Myers, "Simple Authentication and Security Layer (SASL)," IETF RFC 2222, October 1997.
104. R. Housley and P. Hoffman, "Internet X.509 Public Key Infrastructure—Operational Protocols: FTP and HTTP)," IETF RFC 2585, May 1999.
105. W. Burr, D. Dodson, N. Nazaria, and W. T. Polk, "MISPC—Minimum Interoperability Specifications for PKI Components," Version 1, September 1997.
106. J. Feigenbaum, "Towards an Infrastructure for Authorization," in *Proceedings of Invited Talks on Public Key Infrastructure, 3rd USENIX Workshop on Electronic Commerce,* pp. 15–19, August/September 1998.
107. D. W, Chadwick and A. Ottenko, "RBAC Policies in XML for X.509 Based Privilege Management," in *Security in the Information Society: Visions and Perspectives, Proceedings of IFIP/SEC2002,* Kluwer Academic Publishers, Norwell, Massachusetts, pp. 39–53, 2002.
108. N. Simoni and S. Znaty, *Gestion de réseau et de service: Similitude des concepts, spécificité des solutions,* InterEditions (Masson), Paris, 1997.
109. U. Blumenthal and B. Wijnen, "User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)," IETF RFC 2274, January 1998.
110. M. Rozenblit, *Security for Telecommunications Network Management,* IEEE Press, New York, 2000.
111. B. Wijnen, R. Presuhn and K. McCloghrie, "View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMPv3)," IETF RFC 2275, January 1998.
112. H. E. Hia, "Examining How Secured SNMP Network Management Consumes Network Resources," http://filebox.vt.edu/users/hhia/SNMP-IPSec-Tests.htm, 2000.
113. M. Abadi and R. Needham, "Prudent Engineering Practice for Cryptographic Protocols," *IEEE Transactions on Software Engineering,* vol. 22, no. 1, pp. 6–15, 1996.
114. K. Fu, E. Sit, K. Smith, and N. Feamster, "Dos and Don'ts of Client Authentication on the Web," available at http://cookies.lcs.mit.edu, 2001 (a shorter version was published in the *Proceedings of the 10th USENIX Security Symposium,* Washington, D.C., August 2001).
115. B. Schneier, "Why Cryptography is Harder than it Looks," available at http://www.counterpane.com/whycrypt.html, 1996.
116. B. Schneier, "Security piftalls in cryptology," available at http://www.counterpane.com/pitfalls.html, 1998.
117. D. Kristol and L. Montulli, "HTTP State Management Mechanism," IETF RFC 2109, February 1997.
118. ANSI X9.9, "American National Standard—Financial Institution Message Authentication (Wholesale)" (Revision of X9.9-1982), American Bankers Association, 1986.
119. ANSI X9.19, "American National Standard—Financial Institution Retail Message Authentication," American Bankers Association, 1986.
120. ISO 8731-1, *Banking—Approved Algorithms for Message Authentication—Part 1:DEA,* 1987.
121. ISO/IEC 9797, *Information Technology—Security Techniques—Data Integrity Mechanism 21ing a Cryptographic Check Function Employing a Block Cipher Algorithm,* 1993.
122. J. Nechvatal, E. Baker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, *Report on the Development of the Advanced Encryption Standard (AES),* National Institute of Standards and Technology, October 2000.

123. D. Boneh and H. Shacham, "Fast Variants of RSA," *CryptoBytes,* vol. 5, no. 1, pp. 1–9, 2002; available at http://www.rsa.com/rsalabs/pubs/cryptobytes.html.

124. M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption—How to Encrypt with RSA," in *Advances in Cryptology—Eurocrypt'94, Workshop on the Theory and Application of Cryptographic Techniques,* A. DeSantis, Ed., *Lecture Notes in Computer Science,* Springer-Verlag, Berlin, pp. 92–111, 1995.

125. B. Kaliski, "PKCS #7: Cryptographic Message Syntax Version 1.5," IETF RFC 2315, March 1998.

126. D. Steedman, *Abstract Syntax Notation One ASN.1: The Tutorial and Reference,* Technology Appraisals, Twickenham, U.K., 1993.

127. ITU, *Specifications of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1),* Recommendation X.209, 1988.

128. M. Nystrom and B. Kaliski, "PKCS #10: Certification Request Syntax Specification, Version 1.7," IETF RFC 2986, November 2000.

129. D. Atkins, W. Stallings, and P. Zimmerman, "PGP Message Exchange Formats," IETF RFC 1991, August 1996.

130. A. Menezes, *Elliptic Curve Public Key Cryptosystems,* Kluwer Academic Publishing, Boston, 1993.

131. ANSI X9.62, "American National Standard for Financial Services- Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)," American Bankers Association, 1998.

132. M. J. Wiener, "Performance Comparison of Public-key Cryptosystems," *CryptoBytes,* vol. 4, no. 1, pp. 1–5, 1998; available at http://www.rsa.com/rsalabs/pubs/cryptobytes.html.

133. G. B. Agnew, "Cryptography, Data Security, and Applications to C-commerce," in *Electronic Commerce Technology Trends: Challenges and Opportunities,* W. Kou and Y. Yesha, Eds., IBM Press, pp. 69–85, 2000.