TABLE 14.10 IDEA formation of Subkeys

128-bit key (divided into eight 16-bit subkeys)	Bit string from which keys are derived
$Z_1^1\ Z_2^1\ Z_3^1\ Z_4^1\ Z_5^1\ Z_6^1\ Z_1^2\ Z_2^2$	K_0 = Original 128-bit key
$Z_3^2\ Z_4^2\ Z_5^2\ Z_6^2\ Z_1^3\ Z_2^3\ Z_3^3\ Z_4^3$	K_1 = 25-bit rotation of K_0
$Z_5^3\ Z_6^3\ Z_1^4\ Z_2^4\ Z_3^4\ Z_4^4\ Z_5^4\ Z_6^4$	K_2 = 25-bit rotation of K_1
$Z_1^5\ Z_2^5\ Z_3^5\ Z_4^5\ Z_5^5\ Z_6^5\ Z_1^6\ Z_2^6$	K_3 = 25-bit rotation of K_2
$Z_3^6\ Z_4^6\ Z_5^6\ Z_6^6\ Z_1^7\ Z_2^7\ Z_3^7\ Z_4^7$	K_4 = 25-bit rotation of K_3
$Z_5^7\ Z_6^7\ Z_1^8\ Z_2^8\ Z_3^8\ Z_4^8\ Z_5^8\ Z_6^8$	K_5 = 25-bit rotation of K_4
$Z_1^{out}\ Z_2^{out}\ Z_3^{out}\ Z_4^{out}$	First 64 bits of K_6 where K_6 = 25-bit rotation of K_5

swapped. After the completion of the eighth round, the four subblocks are manipulated in a final output transformation. For the representation of $Z_x^{(R)}$ shown in Tables 14.10 and 14.11, the round number is shown without parentheses for ease of notation.

Each round consists of the steps shown in Table 14.12. The final values from steps 11–14 form the output of the round. The two inner 16-bit data subblocks (except for the last round) are swapped, and then these four subblocks are the input to the next round. This technique continues for a total of 8 rounds. After round 8, the final output transformation is as follows:

1. $M_1 \times Z_1^{out}$ (first subkey of output transformation)
2. $M_2 + Z_2^{out}$
3. $M_3 + Z_3^{out}$
4. $M_4 \times Z_4^{out}$

TABLE 14.11 IDEA Subkey Schedule

Round	Set of Encryption Subkeys	Set of Decryption Subkeys
1	$Z_1^1\ Z_2^1\ Z_3^1\ Z_4^1\ Z_5^1\ Z_6^1$	$(Z_1^{out})^{-1} - Z_2^{out} - Z_3^{out}\ (Z_4^{out})^{-1}\ Z_5^8\ Z_6^8$
2	$Z_1^2\ Z_2^2\ Z_3^2\ Z_4^2\ Z_5^2\ Z_6^2$	$(Z_1^8)^{-1} - Z_2^8 - Z_3^8\ (Z_4^8)^{-1}\ Z_5^7\ Z_6^7$
3	$Z_1^3\ Z_2^3\ Z_3^3\ Z_4^3\ Z_5^3\ Z_6^3$	$(Z_1^7)^{-1} - Z_2^7 - Z_3^7\ (Z_4^7)^{-1}\ Z_5^6\ Z_6^6$
4	$Z_1^4\ Z_2^4\ Z_3^4\ Z_4^4\ Z_5^4\ Z_6^4$	$(Z_1^6)^{-1} - Z_2^6 - Z_3^6\ (Z_4^6)^{-1}\ Z_5^5\ Z_6^5$
5	$Z_1^5\ Z_2^5\ Z_3^5\ Z_4^5\ Z_5^5 Z_6^5$	$(Z_1^5)^{-1} - Z_2^5 - Z_3^5\ (Z_4^5)^{-1}\ Z_5^4\ Z_6^4$
6	$Z_1^6\ Z_2^6\ Z_3^6\ Z_4^6\ Z_5^6\ Z_6^6$	$(Z_1^4)^{-1} - Z_2^4 - Z_3^4\ (Z_4^4)^{-1}\ Z_5^3\ Z_6^3$
7	$Z_1^7\ Z_2^7\ Z_3^7\ Z_4^7\ Z_5^7\ Z_6^7$	$(Z_1^3)^{-1}\ Z_2^3 - Z_3^3\ (Z_4^3)^{-1}\ Z_5^2\ Z_6^2$
8	$Z_1^8\ Z_2^8\ Z_3^8\ Z_4^8\ Z_5^8\ Z_6^8$	$(Z_1^2)^{-1} - Z_2^2 - Z_3^2\ (Z_4^2)^{-1}\ Z_5^1\ Z_6^1$
Output Transformation	$Z_1^{out}\ Z_2^{out}\ Z_3^{out}\ Z_4^{out}$	$(Z_1^1)^{-1} - Z_2^1 - Z_3^1\ (Z_4^1)^{-1}$

Example 14.8 The First Round of the IDEA Cipher

Consider that the message is the word "HI", which we first transform to hexadecimal (hex) notation. We start with the ASCII code table in Figure 2.3, where bit 1 is the least significant bit (LSB). We then add an eighth zero-value most significant bit (MSB), which might ordinarily be used for parity, and we transform four bits at a time reading from MSB to LSB. Thus, the letter H in the message transforms to 0048 and

TABLE 14.12 IDEA Operational Steps in Each Round

1. $M_1 \times Z_1^{(R)}$
2. $M_2 + Z_2^{(R)}$
3. $M_3 + Z_3^{(R)}$
4. $M_4 \times Z_4^{(R)}$
5. XOR[3] the results from steps 1 and 3.
6. XOR the results from steps 2 and 4.
7. Result from step 5 and $Z_5^{(R)}$ are multiplied.
8. Result from step 6 and 7 are added.
9. Result from step 8 and $Z_6^{(R)}$ are multiplied.
10. Results from steps 7 and 9 are added.
11. XOR the results from steps 1 and 9.
12. XOR the results from steps 3 and 9.
13. XOR the results from steps 2 and 10.
14. XOR the results from steps 4 and 10.

[3] The exclusive-OR (XOR) operation is defined as: 0 XOR 0 = 0, 0 XOR 1 = 1, 1 XOR 0 = 1, and 1 XOR 1 = 0

the letter I transforms to 0049. For this example, we choose a 128-bit key, K_0, expressed with eight groups or *subkeys* of 4-hex digits each, as follows: K_0 = 0008 0007 0006 0005 0004 0003 0002 0001, where the rightmost subkey is the least significant. Using this key and the IDEA cipher, find the output of round 1.

Solution

The message is first divided into 64-bit data blocks. Each of these blocks is then divided into subblocks, M_i, where $i = 1,\ldots 4$, each subblock containing 16-bits or 4-hex digits. In this example the message "HI" is only 16-bits in length, hence (using hex notation) M_1 = 4849 and $M_2 = M_3 = M_4$ = 0000. Addition is performed modulo 2^{16}, and multiplication is performed modulo $2^{16} + 1$. For the first round, the specified 128-bit key is divided into eight 16-bit subkeys starting with the least significant group of hex digits, as follows: $Z_1^{(1)}$ = 0001, $Z_2^{(1)}$ = 0002, $Z_3^{(1)}$ = 0003, $Z_4^{(1)}$ = 0004, $Z_5^{(1)}$ = 0005, $Z_6^{(1)}$ = 0006, $Z_1^{(2)}$ = 0007, and $Z_2^{(2)}$ = 0008.

The steps outlined in Table 14.11 yield:

1. $M_1 \times Z_1 = 4849 \times 0001 = 4849$.
2. $M_2 + Z_2 = 0000 + 0002 = 0002$.
3. $M_3 + Z_3 = 0000 + 0003 = 0003$.
4. $M_4 \times Z_4 = 0000 \times 0004 = 0000$.
5. The result from step (1) is XOR'ed with the result from step (3) yielding 4849 XOR 0003 = 484A, as follows:

```
      0100  1000  0100  1001   (4849 hex converted to binary)
XOR   0000  0000  0000  0011   (0003 hex converted to binary)
      ---------------------
      0100  1000  0100  1010
```

Converting back to hex yields: 484A (where A is the hex notation for 1010 binary)

6. Results from steps (2) and (4) are XOR'ed: 0002 XOR 0000 = 0002.
7. Result from step (5) and Z_5 are multiplied: 484A × 0005 = 6971.
8. Results from steps (6) and (7) are added: 0002 + 6971 = 6973.

9. Result from step (8) and Z_6 are multiplied: 6973 × 0006 = 78B0.
10. Results from steps (7) and (9) are added: 6971 + 78B0 = E221.
11. Results from steps (1) and (9) are XOR'ed: 4849 XOR 78B0 = 30F9.
12. Results from steps (3) and (9) are XOR'ed: 0003 XOR 78B0 = 78B3.
13. Results from steps (2) and (10) are XOR'ed: 0002 XOR E221 = E223.
14. Results from steps (4) and (10) are XOR'ed: 0000 XOR E221 = E221.

The output of round 1 (the result from steps 11–14) is: 30F9 78B3 E223 E221. Prior to the start of round 2, the two inner words of the round 1 output are swapped. Then, seven additional rounds and a final output transformation are performed.

14.6.2 Diffie-Hellman (Elgamal Variation) and RSA

For encryption of the session key, PGP offers a choice of two public-key encryption algorithms, RSA and the Diffie-Hellman (Elgamal variation) protocol. PGP allows for key sizes of 1024 to 4096 bits for RSA or Diffie-Hellman algorithms. The key size of 1024 bits is considered safe for exchanging most information. The security of the RSA algorithm (see Section 14.5.3) is based on the difficulty of factoring large integers.

The Diffie-Hellman protocol was developed by Whitfield Diffie, Martin E. Hellman, and Ralph C. Merkle in 1976 [19, 20] for public-key exchange over an insecure channel. It is based on the difficulty of the discrete logarithm problem for finite fields [21]. It assumes that it is computationally infeasible to compute g^{ab} knowing only g^a and g^b. U.S. Patent 4,200,770, which expired in 1997, covers the Diffie-Hellman protocol and variations such as Elgamal. The Elgamal variation, which was developed by Taher Elgamal, extends the Diffie-Hellman protocol for message encryption. PGP employs the Elgamal variation of Diffie-Hellman for the encryption of the session-key.

14.6.2.1 Description of Diffie-Hellman, Elgamal Variant:

The protocol has two-system parameter n and g that are both public. Parameter n is a large prime number, and parameter g is an integer less than n that has the following property: for every number p between 1 and $n - 1$ inclusive, there is a power k of g such that $g^k = p \bmod n$. The Elgamal encryption scheme [19, 21] that allows user B to send a message to user A is described below:

- User A randomly chooses a large integer, a (this is user A's private key).
- User A's public key is computed as: $y = g^a \bmod n$.
- User B wishes to send a message M to user A. User B first generates a random number k that is less than n.
- User B computes the following:

$$y_1 = g^k \bmod n$$

$$y_2 = M \times (y^k \bmod n)$$ (recall that y is users A's public key).

- User B sends the ciphertext (y_1, y_2) to user A.

- Upon receiving ciphertext (y_1, y_2), user A computes the plaintext message M as follows:

$$M = \frac{y_2}{y_1^a \bmod n}$$

Example 14.9 Diffie-Hellman (Elgamal variation) for Message Encryption

Consider that the public-system parameters are $n = 11$ and $g = 7$. Suppose that user A chooses the private key to be $a = 2$. Show how user A's public key is computed. Also, show how user B would encrypt a message $M = 13$ to be sent to user A, and how user A subsequently decrypts the ciphertext to yield the message.

Solution

User A's public key $(y = g^a \bmod n)$ is computed as: $y = 7^2 \bmod 11 = 5$. User B wishes to send message $M = 13$ to user A. For this example, let user B randomly choose a value of k (less than $n = 11$) to be $k = 1$. User B computes the ciphertext pair

$$y_1 = g^k \bmod n = 7^1 \bmod 11 = 7$$

$$y_2 = M \times (y^k \bmod n) = 13 \times (5^1 \bmod 11) = 13 \times 5 = 65$$

User A receives the ciphertext (7, 65), and computes message M as follows:

$$M = \frac{y_2}{y_1^a \bmod n} = \frac{65}{7^2 \bmod 11} = \frac{65}{5} = 13$$

14.6.3 PGP Message Encryption

The private-key algorithms that PGP uses for message encryption were presented in Section 14.6.1. The public-key algorithms that PGP uses to encrypt the private-session key were presented in Section 14.6.2. The next example combines the two types of algorithms to illustrate the PGP encryption technique shown in Figure 14.20.

Example 14.10 PGP Use of RSA and IDEA for Encryption

For the encryption of the session key, use the RSA public-key algorithm with the parameters taken from Section 14.5.3.1, where $n = pq = 2773$, the encryption key is $e = 17$, and the decryption key is $d = 157$. The encryption key is the recipient's public key, and the decryption key is the recipient's private key. From Example 14.8, use the session key $K_0 = 0008\ 0007\ 0006\ 0005\ 0004\ 0003\ 0002\ 0001$, and the ciphertext of 30F9 78B3 E223 E221 representing the message "HI", where all the digits are shown in hexadecimal notation. (Note that the ciphertext was created by using only one round of the IDEA algorithm. In the actual implementation, 8 rounds plus an output transformation are performed.) Encrypt the session key, and show the PGP transmission that would be made.

Solution

Following the description in Section 14.5.3.1, the session key will be encrypted using the RSA algorithm with the recipient's public key of 17. For ease of calculation with a simple calculator, let us first transform the session key into groups made up of base-10 digits. In keeping with the requirements of the RSA algorithm, the value ascribed to any group may not exceed $n - 1 = 2772$. Therefore, let us express the 128-bit key in terms of 4-digit groups, where we choose the most significant group (leftmost) to represent 7 bits, and the balance of the 11 groups to represent 11 bits each. The transfor-

mation from base-16 to base-10 digits can best be viewed as a two-step process, (1) conversion to binary and, (2) conversion to base 10. The result is K_0 = 0000 0032 0000 1792 0048 0001 0512 0064 0001 1024 0064 0001. Recall from Equation 14.32, that $C = (M)^e$ modulo-n where M will be one of the 4-digit groups of K_0. The leftmost four groups are encrypted as:

$$C_{12} = (0000)^{17} \bmod 2773 = 0.$$
$$C_{11} = (0032)^{17} \bmod 2773 = 2227.$$
$$C_{10} = (0000)^{17} \bmod 2773 = 0.$$
$$C_9 = (1792)^{17} \bmod 2773 = 2704.$$

An efficient way to compute modular exponentiation is to use the Square-and-Multiply algorithm. This algorithm [21] reduces the number of modular multiplications needed to be performed from $e - 1$ to at most 2ℓ, where ℓ is the number of bits in the binary representation. Let us demonstrate the use of the Square-and-Multiply algorithm by encrypting one of the session-key decimal groups (the eleventh group from the right, M_{11} = 0032), where n = 2773 and e = 17. In using this algorithm, we first convert e to its binary representation (17 decimal = 10001 binary).

The calculations are illustrated in Table 14.13. Modulo-n math is used, where n = 2773 in this example. The second column contains the binary code, with the most significant bit (MSB) in row 1. Each bit value in this column acts to control a result in column 3. The starting value, placed in column 3 row 0, is always 1. Then, the result for any row in column 3 depends on the value of the bit in the corresponding row in column 2: if that entry contains a "1," then the previous row-result is squared and multiplied by the plaintext (32 for this example). If a row in the second column contains a "0", then the result of that row in column 3 equals only the square of the previous row's result. The final value is the encrypted ciphertext (C = 2227). Repeating this method for each of the twelve decimal groups that comprise K_0 results in the ciphertext of the session key to be: C = 0000 2227 0000 2704 0753 0001 1278 0272 0001 1405 0272 0001. This RSA-encrypted session key (represented here in decimal) together with the IDEA-encrypted message of 30F9 78B3 E223 E221 (represented here in hex) can now be transmitted over an insecure channel.

TABLE 14.13 The Square-and-Multiply Algorithm with Plaintext = 32

Row Number	Binary representation of e (MSB first)	Modulo multiplication (modulo 2773)
0		1
1	1	$1^2 \times 32 = 32$
2	0	$32^2 = 1024$
3	0	$1024^2 = 382$
4	0	$382^2 = 1728$
5	1	$1728^2 \times 32 = 2227$

14.6.4 PGP Authentication and Signature

The public key algorithms can be used to authenticate or "sign" a message. As illustrated in Figure 14.18, a sender can encrypt a document with his private key (which no one else has access to) prior to encrypting it with the recipient's public

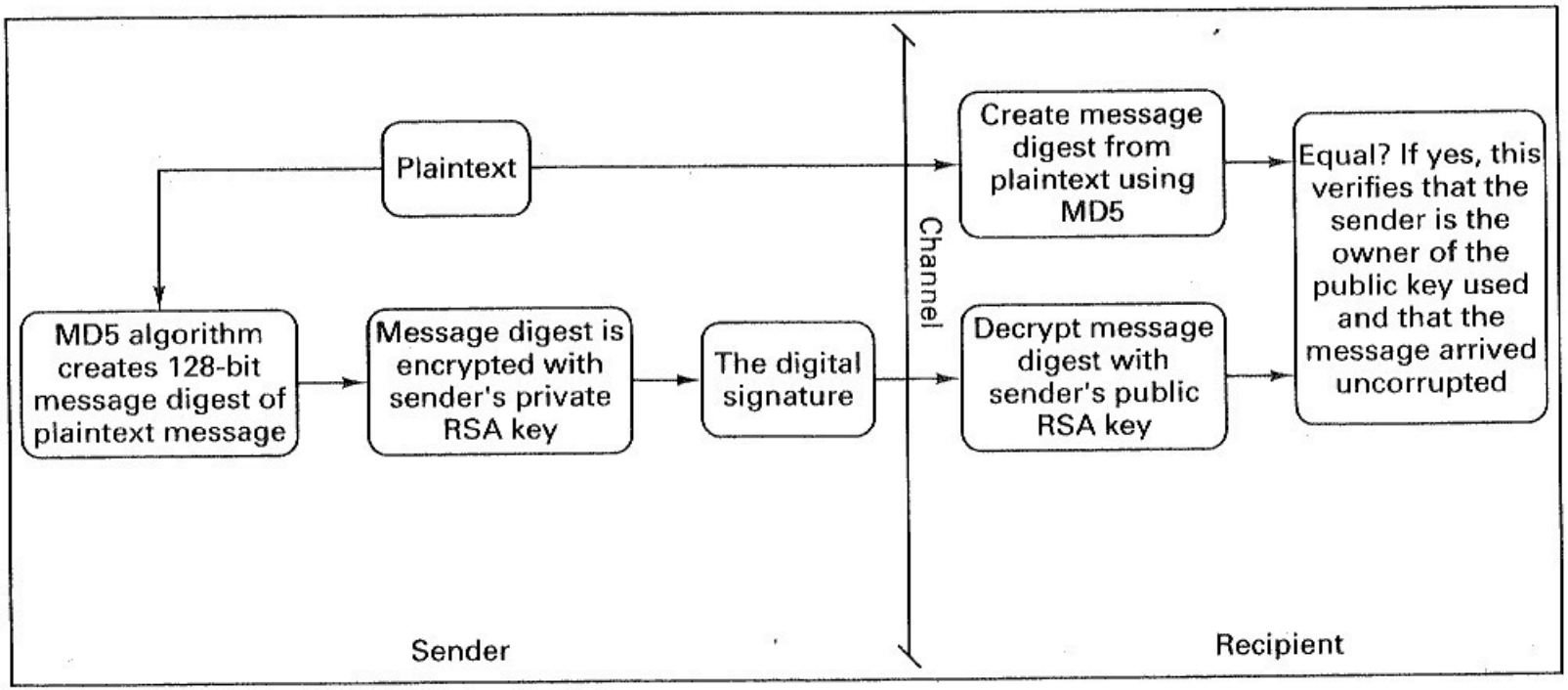

Figure 14.22 PGP signature technique.

key. The recipient must first use his private key to decrypt the message, followed by a second decryption using the sender's public key. This technique encrypts the message for secrecy and also provides authentication of the sender.

Because of the slowness of public-key algorithms, PGP allows for a different method of authenticating a sender. Instead of the time-consuming process of encrypting the entire plaintext message, the PGP approach encrypts a fixed-length message digest created with a one-way hash function. The encryption of the message digest is performed using a public-key algorithm. This method is known as a *digital signature* and is shown in Figure 14.22. A digital signature is used to provide authentication of both *the sender* and *the message.* Authentication of the message provides a verification that the message was not altered in some way. Using this technique, if a message has been altered in any way (i.e. by a forger), its message digest will be different.

PGP version 2.6 uses the MD5 (Message Digest 5) algorithm to create a 128-bit message digest (or hash value) of the plaintext. This hash value is then encrypted with the sender's private key and sent with the plaintext. When the recipient receives the message, he will first decrypt the message digest with the sender's public key. The recipient will then apply the hash function to the plaintext and compare the two message digests. If they match, the signature is valid. In Figure 14.22, the message is sent without encryption (as plaintext), but it may be encrypted by the method illustrated in Figure 14.20.

14.6.4.1 MD5 and SHA-1

MD5 and SHA-1 are hash functions. A hash function $H(x)$ takes an input and returns a fixed-size string h, called the hash value (also known as a message digest). A cryptographic hash function has the following properties:

1. The output length is fixed.
2. The hash value is relatively simple to compute.
3. The function is one way—in other words, it is hard to invert. If given a hash value h, it is computationally infeasible to find the function's input x.
4. The function is *collision free.* A collision-free hash function is a function for which it is infeasible that two different messages will create the same hash value.

The MD5 algorithm used in PGP version 2.6 creates a 128-bit message digest. The MD5 algorithm processes the text in 512-bit blocks through four rounds of data manipulation. Each round uses a different nonlinear function that consists of the logical operators AND, OR, NOT or XOR. Each function is performed 16 times in a round. Bit shifts and scalar additions are also performed in each round [19]. Hans Dobbertin [18] has determined that collisions may exist in MD5. Because of this potential weakness, the PGP specification recommends using the Digital Signature Standard (DSS). DSS uses the SHA-1 (Secure Hash Algorithm-1) algorithm. The SHA-1 algorithm takes a message of less than 2^{64} bits in length and produces a 160-bit

message digest. SHA-1 is similar to MD5 in that it uses a different nonlinear function in each of its 4 rounds. In SHA-1, each function is performed 20 times per round. SHA-1 also uses various scalar additions and bit shifting. The algorithm is slightly slower than MD5 but the larger message digest (160-bit versus 128 bit) makes it more secure against brute-force attacks [19]. A brute-force attack consists of trying many input combinations in an attempt to match the message digest under attack.

14.6.4.2 Digital Signature Standard and RSA

For digital signatures, PGP version 2.6 uses the RSA algorithm for encryption of the hash value produced by the MD5 function; however, versions 5.0 and later adhere to the NIST Digital Signature Standard (DSS) [22]. The NIST DSS requires the use of the SHA-1 hash function. The hash value is then encrypted using the Digital Standard Algorithm (DSA). Like the Diffie-Hellman protocol, DSA is based on the discrete logarithm problem. (Reference [22] contains a detailed description of DSA.)

14.7 CONCLUSION

In this chapter we have presented the basic model and goals of the cryptographic process. We looked at some early cipher systems and reviewed the mathematical theory of secret communications established by Shannon. We defined a system that can exhibit perfect secrecy and established that such systems can be implemented but that they are not practical for use where high-volume communications are required. We also considered practical security systems that employ Shannon's techniques (known as confusion and diffusion) to frustrate the statistical endeavors of a cryptanalyst.

The outgrowth of Shannon's work was utilized by IBM in the LUCIFER system, which later grew into the National Bureau of Standards' Data Encryption Standard (DES). We outlined the DES algorithm in detail. We also considered the use of linear feedback shift registers (LFSR) for stream encryption systems, and demonstrated the intrinsic vulnerability of an LFSR used as a key generator.

We also looked at the area of public-key cryptosystems and examined two schemes, the Rivest–Shamir–Adelman (RSA) scheme, based on the product of two large prime numbers, and the Merkle-Hellman scheme, based on the classical knapsack problem. Finally, we looked at the novel scheme of Pretty Good Privacy (PGP), developed by Phil Zimmerman and published in 1991. PGP utilizes the benefits of both private and public-key systems and has proven to be an important file-encryption method for sending data via electronic mail.

REFERENCES

1. Kahn, D., *The Codebreakers*, Macmillan Publishing Company, New York, 1967.
2. Diffie, W., and Hellman, M.E., "Privacy and Authentication: An Introduction to Cryptography," *Proc. IEEE*, vol. 67, no. 3, Mar. 1979, pp. 397–427.

3. Beker,H., and Piper, F., *Cipher Systems,* John Wiley & Sons, Inc., New York, 1982.
4. Denning, D.E.R., *Cryptography and Data Security,* Addison-Wesley Publishing Company, Reading, Mass., 1982.
5. Shannon, C.E., "Communication Theory of Secrecy Systems," *Bell Syst. Tech. J.,* vol. 28, Oct. 1949, pp. 656–715.
6. Hellman, M. E., "An Extension of the Shannon Theory Approach to Cryptography," *IEEE Trans. Inf. Theory,* vol. IT23, May 1978, pp. 289–294.
7. Smith, J. L., "The Design of Lucifer, a Cryptographic Device for Data Communications," *IBM Research Rep. RC-3326,* 1971.
8. Feistel, H. "Cryptography and Computer Privacy," *Sci. Am.,* vol. 228, no. 5, May 1973, pp. 15–23.
9. National Bureau of Standards, "Data Encryption Standard," *Federal Information Processing Standard (FIPS),* Publication no. 46, Jan. 1977.
10. United States Senate Select Committee on Intelligence, "Unclassified Summary: Involvement of NSA in the Development of the Data Encryption Standard," *IEEE Commun. Soc. Mag.,* vol. 16, no. 6, Nov. 1978, pp. 53–55.
11. Stallings, W., *Cryptography and Network Security, Second Edition,* Prentice Hall, Upper Saddle River, NJ. 1998.
12. Diffie, W., and Hellman, M. E., "New Directions in Cryptography," *IEEE Trans. Inf. Theory,* vol. IT22, Nov. 1976. pp. 644–654.
13. Rivest, R.L., Shamir, A., and Adelman, L., "On Digital Signatures and Public Key Cryptosystems," *Commun. ACM,* vol. 21, Feb. 1978, pp. 120–126.
14. Knuth, D. E., *The Art of Computer Programming,* Vol. 2, *Seminumerical Algorithms,* 2nd ed., Addison-Wesley Publishing Company, Reading, Mass., 1981.
15. Merkel, R. C., and Hellman, M. E., "Hiding Information and Signatures in Trap-Door Knapsacks," *IEEE Trans. Inf. Theory,* vol. IT24, Sept. 1978, pp. 525–530.
16. Shamir, A., "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," *IEEE 23rd Ann. Symp. Found. Comput. Sci.,* 1982, pp. 145–153.
17. Zimmerman, P. *The Official PGP User's Guide,* MIT Press, Cambridge, 1995.
18. *PGP Freeware User's Guide, Version 6.5,* Network Associates, Inc., 1999.
19. Schneier, B., *Applied Cryptography,* John Wiley & Sons, New York, 1996.
20. Hellman, M. E., Martin, Bailey, Diffie, W., and Merkle, R. C., *United States Patent 4,200,700: Cryptographic Apparatus and Method,* United States Patent and Trademark Office, Washington, DC, 1980.
21. Stinson, Douglas, *Cryptography Theory and Practice.* CRC Press, Boca Raton, FL, 1995.
22. *Digital Signature Standard* (Federal Information Processing Standards Publication 186–1), Government Printing Office, Springfield, VA, Dec. 15, 1998.

PROBLEMS

14.1. Let X be an integer variable represented with 64 bits. The probability is $\frac{1}{2}$ that X is in the range $(0, 2^{16} - 1)$, the probability is $\frac{1}{4}$ that X is in the range $(2^{16}, 2^{32} - 1)$, and the probability is $\frac{1}{4}$ that X is in the range $(2^{32}, 2^{64} - 1)$. Within each range the values are equally likely. Compute the entropy of X.

14.2. A set of equally likely weather messages are: sunny (S), cloudy (C), light rain (L), and heavy rain (H). Given the added information concerning the time of day (morning or afternoon), the probabilities change as follows:

$$\text{Morning:} \quad P(S) = \tfrac{1}{8}, \quad P(\text{C}) = \tfrac{1}{8}, \quad P(\text{L}) = \tfrac{3}{8}, \quad P(\text{H}) = \tfrac{3}{8}$$

$$\text{Afternoon:} \quad P(\text{S}) = \tfrac{3}{8}, \quad P(\text{C}) = \tfrac{3}{8}, \quad P(\text{L}) = \tfrac{1}{8}, \quad P(\text{H}) = \tfrac{1}{8}$$

(a) Find the entropy of the weather message.
(b) Find the entropy of the message conditioned on the time of day.

14.3. The Hawaiian alphabet has only 12 letters—the vowels, a, e, i, o, u, and the consonants, h, k, l, m, n, p, w. Assume that each vowel occurs with probability 0.116, and that each consonant occurs with probability 0.06. Also assume that the average number of *information bits* per letter is the same as that for the English language. Calculate the unicity distance for an encrypted Hawaiian message if the key sequence consists of a random permutation of the 12-letter alphabet.

14.4. Estimate the unicity distance for an English language encryption system that uses a key sequence made up of 10 random alphabetic characters:
(a) Where each key character can be any one of the 26 letters of the alphabet (duplicates are allowed).
(b) Where the key characters may not have any duplicates.

14.5. Repeat Problem 14.4 for the case where the key sequence is made up of ten integers randomly chosen from the set of numbers 0 to 999.

14.6. **(a)** Find the unicity distance for a DES system which encrypts 64-bit blocks (eight alphabetic characters) using a 56-bit key.
(b) What is the effect on the unicity distance in part (a) if the key is increased to 128 bits?

14.7. In Figures 14.8 and 14.9, *P*-boxes and *S*-boxes alternate. Is this arrangement any more secure than if all the *P*-boxes were first grouped together, followed by all the *S*-boxes similarly grouped together? Justify your answer.

14.8. What is the output of the first iteration of the DES algorithm when the plaintext and the key are each made up of zero sequences?

14.9. Consider the 10-bit plaintext sequence 0 1 0 1 1 0 1 0 0 1 and its corresponding ciphertext sequence 0 1 1 1 0 1 1 0 1 0, where the rightmost bit is the earliest bit. Describe the five-stage linear feedback shift register (LFSR) that produced the key sequence and show the initial state of the register. Is the output sequence of maximal length?

14.10. Following the RSA algorithm and parameters in Example 14.5, compute the encryption key, e, when the decryption key is chosen to be 151.

14.11. Given e and d that satisfy ed modulo-$\phi(n) = 1$, and a message that is encoded as an integer number, M, in the range $(0, n-1)$ such that the gcd $(M, n) = 1$. Prove that $(M^e \text{ modulo-}n)^d \text{ modulo-}n = M$.

14.12. Use the RSA scheme to encrypt the message $M = 3$. Use the prime numbers $p = 5$ and $q = 7$. Choose the decryption key, d, to be 11, and calculate the value of the encryption key, e.

14.13. Consider the following for the RSA scheme.
(a) If the prime numbers are $p = 7$ and $q = 11$, list five allowable values for the decryption key, d.

(b) If the prime numbers are $p = 13$, $q = 31$, and the decryption key is $d = 37$, find the encryption key, e, and describe how you would use it to encrypt the word "DIGITAL."

14.14. Use the Merkle–Hellman public key scheme with the super-increasing vector, $\mathbf{a}' = 1, 3, 5, 10, 20$. Use the following additional parameters: a large prime number $M = 51$ and a random number $W = 37$.

(a) Find the nonsuper-increasing vector, **a**, to be made public, and encrypt the data vector 1 1 0 1 1.

(b) Show the steps by which an authorized receiver decrypts the ciphertext.

14.15. Using the Diffie-Hellman (Elgamal variation) protocol, encrypt the message $M = 7$. The system parameters are $n = 17$ and $g = 3$. The recipients private key is $a = 4$. Determine the recipient's public key. For message encryption with the randomly selected k, use $k = 2$. Verify the accuracy of the ciphertext by performing decryption using the recipient's private key.

14.16. Find the hexadecimal (hex) value of the message "no" after one round of the IDEA algorithm. The session key in hex notation is = 0002 0003 0002 0003 0002 0003 0002 0003, where the rightmost 4-digit group represents the subkey Z_1. For the message "no," let each ASCII character be represented by a 16-bit data subblock, where "n" = 006E and "o" = 006F.

14.17. In the PGP Example 14.10, the IDEA session key is encrypted using the RSA algorithm. The resulting encrypted session key (in base-10 notation) was: 0000 2227 0000 2704 0753 0001 1278 0272 0001 1405 0272 0001, where the least significant (rightmost) group is group 1. Using the decryption key, decrypt group 11 of this session key using the Square-and-Multiply technique.

QUESTIONS

14.1. What are the two major requirements for a useful *cryptosystem*? (See Section 14.1.2.)

14.2. Shannon suggested two encryption concepts that he termed *confusion* and *diffusion*. Explain what these terms mean. (See Section 14.3.1.)

14.3. If *high-level security* is desired, explain why a linear feedback shift register (LFSR) would not be used. (See Section 14.4.2.)

14.4. Explain the major difference between conventional cryptosystems and *public key cryptosystems*. (See Section 14.5.)

14.5. Describe the steps used for message encryption employed by the *Data Encryption Standard* (DES). How different is the operation when using Triple-DES? (See Sections 14.3.5 and 14.6.1.1)

14.6. Describe the steps used for message encryption employed by version 2.6 of the *Pretty Good Privacy* (PGP) technique. (See Section 14.6.1.3.)

EXERCISES

Using the Companion CD, run the exercises associated with Chapter 14.