SOLUTIONS MANUAL

CRYPTOGRAPHY AND NETWORK SECURITY

PRINCIPLES AND PRACTICE

FOURTH EDITION

WILLIAM STALLINGS

NOTICE

This manual contains solutions to all of the review questions and homework problems in *Cryptography and Network Security, Fourth Edition*. If you spot an error in a solution or in the wording of a problem, I would greatly appreciate it if you would forward the information via email to ws@shore.net. An errata sheet for this manual, if needed, is available at ftp://shell.shore.net/members/w/s/ws/S.

W.S.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

ANSWERS TO QUESTIONS

1.1 The OSI Security Architecture is a framework that provides a systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. The document defines security attacks, mechanisms, and services, and the relationships among these categories.

1.2 **Passive attacks** have to do with eavesdropping on, or monitoring, transmissions. Electronic mail, file transfers, and client/server exchanges are examples of transmissions that can be monitored. **Active attacks** include the modification of transmitted data and attempts to gain unauthorized access to computer systems.

1.3 **Passive attacks:** release of message contents and traffic analysis. **Active attacks:** masquerade, replay, modification of messages, and denial of service.

1.4 **Authentication:** The assurance that the communicating entity is the one that it claims to be.
Access control: The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).
Data confidentiality: The protection of data from unauthorized disclosure.
Data integrity: The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
Nonrepudiation: Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
Availability service: The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system (i.e., a system is available if it provides services according to the system design whenever users request them).

1.5 See Table 1.3.

ANSWERS TO PROBLEMS

1.1	Release of message contents	Traffic analysis	Masquerade	Replay	Modification of messages	Denial of service
Peer entity authentication			Y			
Data origin authentication			Y			
Access control			Y			
Confidentiality	Y					
Traffic flow confidentiality		Y				
Data integrity				Y	Y	
Non-repudiation			Y			
Availability						Y

1.2	Release of message contents	Traffic analysis	Masquerade	Replay	Modification of messages	Denial of service
Encipherment	Y					
Digital signature			Y	Y	Y	
Access control	Y	Y	Y	Y		Y
Data integrity				Y	Y	
Authentication exchange	Y		Y	Y		Y
Traffic padding		Y				
Routing control	Y	Y				Y
Notarization			Y	Y	Y	

CHAPTER 2
CLASSICAL ENCRYPTION TECHNIQUESR

ANSWERS TO QUESTIONS

2.1 Plaintext, encryption algorithm, secret key, ciphertext, decryption algorithm.

2.2 Permutation and substitution.

2.3 One key for symmetric ciphers, two keys for asymmetric ciphers.

2.4 A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

2.5 Cryptanalysis and brute force.

2.6 **Ciphertext only**. One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical. Thus, the opponent must rely on an analysis of the ciphertext itself, generally applying various statistical tests to it. **Known plaintext.** The analyst may be able to capture one or more plaintext messages as well as their encryptions. With this knowledge, the analyst may be able to deduce the key on the basis of the way in which the known plaintext is transformed. **Chosen plaintext.** If the analyst is able to choose the messages to encrypt, the analyst may deliberately pick patterns that can be expected to reveal the structure of the key.

2.7 An encryption scheme is **unconditionally secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available. An encryption scheme is said to be **computationally secure** if: (1) the cost of breaking the cipher exceeds the value of the encrypted information, and (2) the time required to break the cipher exceeds the useful lifetime of the information.

2.8 The **Caesar cipher** involves replacing each letter of the alphabet with the letter standing k places further down the alphabet, for k in the range 1 through 25.

2.9 A **monoalphabetic substitution cipher** maps a plaintext alphabet to a ciphertext alphabet, so that each letter of the plaintext alphabet maps to a single unique letter of the ciphertext alphabet.

2.10 The **Playfair algorithm** is based on the use of a 5×5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

2.11 A **polyalphabetic substitution cipher** uses a separate monoalphabetic substitution cipher for each successive letter of plaintext, depending on a key.

2.12 **1.** There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.
2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

2.13 A **transposition cipher** involves a permutation of the plaintext letters.

2.14 Steganography involves concealing the existence of a message.

ANSWERS TO PROBLEMS

2.1 **a.** No. A change in the value of b shifts the relationship between plaintext letters and ciphertext letters to the left or right uniformly, so that if the mapping is one-to-one it remains one-to-one.

b. 2, 4, 6, 8, 10, 12, 13, 14, 16, 18, 20, 22, 24. Any value of a larger than 25 is equivalent to a mod 26.

c. The values of a and 26 must have no common positive integer factor other than 1. This is equivalent to saying that a and 26 are relatively prime, or that the greatest common divisor of a and 26 is 1. To see this, first note that $E(a, p) = E(a, q)$ $(0 \le p \le q < 26)$ if and only if $a(p - q)$ is divisible by 26. **1.** Suppose that a and 26 are relatively prime. Then, $a(p - q)$ is not divisible by 26, because there is no way to reduce the fraction $a/26$ and $(p - q)$ is less than 26. **2.** Suppose that a and 26 have a common factor $k > 1$. Then $E(a, p) = E(a, q)$, if $q = p + m/k \neq p$.

2.2 There are 12 allowable values of a (1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25). There are 26 allowable values of b, from 0 through 25). Thus the total number of distinct affine Caesar ciphers is $12 \times 26 = 312$.

2.3 Assume that the most frequent plaintext letter is e and the second most frequent letter is t. Note that the numerical values are e = 4; B = 1; t = 19; U = 20. Then we have the following equations:

$1 = (4a + b) \bmod 26$
$20 = (19a + b) \bmod 26$

Thus, $19 = 15a \bmod 26$. By trial and error, we solve: $a = 3$.

Then 1 = (12 + *b*) mod 26. By observation, *b* = 15.

2.4 A good glass in the Bishop's hostel in the Devil's seat—twenty-one degrees and thirteen minutes—northeast and by north—main branch seventh limb east side—shoot from the left eye of the death's head— a bee line from the tree through the shot fifty feet out. (from *The Gold Bug*, by Edgar Allan Poe)

2.5 **a.** The first letter t corresponds to A, the second letter h corresponds to B, e is C, s is D, and so on. Second and subsequent occurrences of a letter in the key sentence are ignored. The result

```
ciphertext:  SIDKHKDM AF HCRKIABIE SHIMC KD LFEAILA
plaintext:   basilisk to leviathan blake is contact
```

b. It is a monalphabetic cipher and so easily breakable.

c. The last sentence may not contain all the letters of the alphabet. If the first sentence is used, the second and subsequent sentences may also be used until all 26 letters are encountered.

2.6 The cipher refers to the words in the page of a book. The first entry, 534, refers to page 534. The second entry, C2, refers to column two. The remaining numbers are words in that column. The names DOUGLAS and BIRLSTONE are simply words that do not appear on that page. Elementary! (from *The Valley of Fear*, by Sir Arthur Conan Doyle)

2.7 **a.**

2	8	10	7	9	6	3	1	4	5
C	R	Y	P	T	O	G	A	H	I
B	E	A	T	T	H	E	T	H	I
R	D	P	I	L	L	A	R	F	R
O	M	T	H	E	L	E	F	T	O
U	T	S	I	D	E	T	H	E	L
Y	C	E	U	M	T	H	E	A	T
R	E	T	O	N	I	G	H	T	A
T	S	E	V	E	N	I	F	Y	O
U	A	R	E	D	I	S	T	R	U
S	T	F	U	L	B	R	I	N	G
T	W	O	F	R	I	E	N	D	S

4	2	8	10	5	6	3	7	1	9
N	E	T	W	O	R	K	S	C	U
T	R	F	H	E	H	F	T	I	N
B	R	O	U	Y	R	T	U	S	T
E	A	E	T	H	G	I	S	R	E
H	F	T	E	A	T	Y	R	N	D

I	R	O	L	T	A	O	U	G	S
H	L	L	E	T	I	N	I	B	I
T	I	H	I	U	O	V	E	U	F
E	D	M	T	C	E	S	A	T	W
T	L	E	D	M	N	E	D	L	R
A	P	T	S	E	T	E	R	F	O

```
ISRNG  BUTLF  RRAFR  LIDLP  FTIYO  NVSEE  TBEHI  HTETA
EYHAT  TUCME  HRGTA  IOENT  TUSRU  IEADR  FOETO  LHMET
NTEDS  IFWRO  HUTEL  EITDS
```

b. The two matrices are used in reverse order. First, the ciphertext is laid out in columns in the second matrix, taking into account the order dictated by the second memory word. Then, the contents of the second matrix are read left to right, top to bottom and laid out in columns in the first matrix, taking into account the order dictated by the first memory word. The plaintext is then read left to right, top to bottom.

c. Although this is a weak method, it may have use with time-sensitive information and an adversary without immediate access to good cryptanalysis (e.g., tactical use). Plus it doesn't require anything more than paper and pencil, and can be easily remembered.

2.8 SPUTNIK

2.9 PT BOAT ONE OWE NINE LOST IN ACTION IN BLACKETT STRAIT TWO MILES SW MERESU COVE X CREW OF TWELVE X REQUEST ANY INFORMATION

2.10 a.

L	A	R	G	E
S	T	B	C	D
F	H	I/J	K	M
N	O	P	Q	U
V	W	X	Y	Z

b.

O	C	U	R	E
N	A	B	D	F
G	H	I/J	K	L
M	P	Q	S	T
V	W	X	Y	Z

2.11 a. UZTBDLGZPNNWLGTGTUEROVLDBDUHFPERHWQSRZ

b. UZTBDLGZPNNWLGTGTUEROVLDBDUHFPERHWQSRZ

c. A cyclic rotation of rows and/or columns leads to equivalent substitutions. In this case, the matrix for part a of this problem is obtained from the matrix of Problem 2.10a, by rotating the columns by one step and the rows by three steps.

2.12 a. $25! \approx 2^{84}$

b. Given any 5x5 configuration, any of the four row rotations is equivalent, for a total of five equivalent configurations. For each of these five configurations, any of the four column rotations is equivalent. So each configuration in fact represents 25 equivalent configurations. Thus, the total number of unique keys is 25!/25 = 24!

2.13 A mixed Caesar cipher. The amount of shift is determined by the keyword, which determines the placement of letters in the matrix.

2.14 a. Difficulties are things that show what men are.

b. Irrationally held truths may be more harmful than reasoned errors.

2.15 a. We need an even number of letters, so append a "q" to the end of the message. Then convert the letters into the corresponding alphabetic positions:

M	e	e	t	m	e	a	t	t	h	e	u	s	u	a	l
13	5	5	20	13	5	1	20	20	8	5	21	19	21	1	12
P	l	a	c	e	a	t	t	e	n	r	a	t	h	e	r
16	12	1	3	5	1	20	20	5	14	18	1	20	8	5	18
T	h	a	n	e	i	g	h	t	o	c	l	o	c	k	q
20	8	1	14	5	9	7	8	20	15	3	12	15	3	11	17

The calculations proceed two letters at a time. The first pair:

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 13 \\ 5 \end{pmatrix} \bmod 26 = \begin{pmatrix} 137 \\ 100 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 \\ 22 \end{pmatrix}$$

The first two ciphertext characters are alphabetic positions 7 and 22, which correspond to GV. The complete ciphertext:

GVUIGVKODZYPUHEKJHUZWFZFWSJSDZMUDZMYCJQMFWWUQRKR

b. We first perform a matrix inversion. Note that the determinate of the encryption matrix is (9 × 7) – (4 × 5) = 43. Using the matrix inversion formula from the book:

$$\begin{pmatrix} 9 & 4 \\ 5 & 7 \end{pmatrix}^{-1} = \frac{1}{43}\begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \bmod 26 = 23\begin{pmatrix} 7 & -4 \\ -5 & 9 \end{pmatrix} \bmod 26 = \begin{pmatrix} 161 & -92 \\ -115 & 9 \end{pmatrix} \bmod 26 = \begin{pmatrix} 5 & 12 \\ 15 & 25 \end{pmatrix}$$

Here we used the fact that $(43)^{-1} = 23$ in Z_{26}. Once the inverse matrix has been determined, decryption can proceed. Source: [LEWA00].

2.16 Consider the matrix **K** with elements k_{ij} to consist of the set of column vectors K_j, where:

$$\mathbf{K} = \begin{pmatrix} k_{11} & \cdots & k_{1n} \\ \vdots & \vdots & \vdots \\ k_{n1} & \cdots & k_{nn} \end{pmatrix} \qquad \text{and} \qquad \mathbf{K}_j = \begin{pmatrix} k_{1j} \\ \vdots \\ k_{nj} \end{pmatrix}$$

The ciphertext of the following chosen plaintext n-grams reveals the columns of **K**:

$$\begin{array}{c} (B, A, A, \ldots, A, A) \leftrightarrow K_1 \\ (A, B, A, \ldots, A, A) \leftrightarrow K_2 \\ : \\ (A, A, A, \ldots, A, B) \leftrightarrow K_n \end{array}$$

2.17 **a.** 7×13^4
b. 7×13^4
c. 13^4
d. 10×13^4
e. $2^4 \times 13^2$
f. $2^4 \times (13^2 - 1) \times 13$
g. 37648
h. 23530
i. 157248

2.18 key: *legleglegle*
plaintext: `explanation`
ciphertext: `PBVWETLXOZR`

2.19 **a.**

s	e	n	d	m	o	r	e	m	o	n	e	y
18	4	13	3	12	14	17	4	12	14	13	4	24
9	0	1	7	23	15	21	14	11	11	2	8	9
1	4	14	10	9	3	12	18	23	25	15	12	7
B	E	C	K	J	D	M	S	X	Z	P	M	H

b.

c	a	s	h	n	o	t	n	e	e	d	e	d
2	0	18	7	13	14	19	13	4	4	3	4	3
25	4	22	3	22	15	19	5	19	21	12	8	4
1	4	14	10	9	3	12	18	23	25	15	12	7
B	E	C	K	J	D	M	S	X	Z	P	M	H

2.20 your package ready Friday 21st room three Please destroy this immediately.

2.21 **a.** Lay the message out in a matrix 8 letters across. Each integer in the key tells you which letter to choose in the corresponding row. Result:

> He sitteth between the cherubims. The isles may be glad thereof. As the rivers in the south.

b. Quite secure. In each row there is one of eight possibilities. So if the ciphertext is 8n letters in length, then the number of possible plaintexts is 8^n.

c. Not very secure. Lord Peter figured it out. (from *The Nine Tailors*)

CHAPTER 3
BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

ANSWERS TO QUESTIONS

3.1 Most symmetric block encryption algorithms in current use are based on the Feistel block cipher structure. Therefore, a study of the Feistel structure reveals the principles behind these more recent ciphers.

3.2 A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.

3.3 If a small block size, such as $n = 4$, is used, then the system is equivalent to a classical substitution cipher. For small n, such systems are vulnerable to a statistical analysis of the plaintext. For a large block size, the size of the key, which is on the order of $n \times 2^n$, makes the system impractical.

3.4 In a product cipher, two or more basic ciphers are performed in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers.

3.5 In **diffusion**, the statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext. This is achieved by having each plaintext digit affect the value of many ciphertext digits, which is equivalent to saying that each ciphertext digit is affected by many plaintext digits. **Confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible, again to thwart attempts to discover the key. Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to make it difficult to deduce the key. This is achieved by the use of a complex substitution algorithm.

3.6 **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed. **Key size:** Larger key size means greater security but may decrease encryption/decryption speed. **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis. **Round function:** Again, greater complexity generally means greater resistance to cryptanalysis. **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern. **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength.

3.7 The S-box is a substitution function that introduces nonlinearity and adds to the complexity of the transformation.

3.8 The avalanche effect is a property of any encryption algorithm such that a small change in either the plaintext or the key produces a significant change in the ciphertext.

3.9 **Differential cryptanalysis** is a technique in which chosen plaintexts with particular XOR difference patterns are encrypted. The difference patterns of the resulting ciphertext provide information that can be used to determine the encryption key. **Linear cryptanalysis** is based on finding linear approximations to describe the transformations performed in a block cipher.

Answers to Problems

3.1 **a.** For an n-bit block size are 2^n possible different plaintext blocks and 2^n possible different ciphertext blocks. For both the plaintext and ciphertext, if we treat the block as an unsigned integer, the values are in the range 0 through $2^n - 1$. For a mapping to be reversible, each plaintext block must map into a unique ciphertext block. Thus, to enumerate all possible reversible mappings, the block with value 0 can map into anyone of 2^n possible ciphertext blocks. For any given mapping of the block with value 0, the block with value 1 can map into any one of $2^n - 1$ possible ciphertext blocks, and so on. Thus, the total number of reversible mappings is $(2^n)!$.

b. In theory, the key length could be $\log_2(2^n)!$ bits. For example, assign each mapping a number, from 1 through $(2^n)!$ and maintain a table that shows the mapping for each such number. Then, the key would only require $\log_2(2^n)!$ bits, but we would also require this huge table. A more straightforward way to define the key is to have the key consist of the ciphertext value for each plaintext block, listed in sequence for plaintext blocks 0 through $2^n - 1$. This is what is suggested by Table 3.1. In this case the key size is $n \times 2n$ and the huge table is not required.

3.2 Because of the key schedule, the round functions used in rounds 9 through 16 are mirror images of the round functions used in rounds 1 through 8. From this fact we see that encryption and decryption are identical. We are given a ciphertext *c*. Let m' = *c*. Ask the encryption oracle to encrypt m'. The ciphertext returned by the oracle will be the decryption of *c*.

3.3 **a.** We need only determine the probability that for the remaining N – t plaintexts P_i, we have $E[K, P_i] \neq E[K', P_i]$. But $E[K, P_i] = E[K', P_i]$ for all the remaining P_i with probability $1 - 1/(N - t)!$.

b. Without loss of generality we may assume the $E[K, P_i] = P_i$ since $E_K(\bullet)$ is taken over all permutations. It then follows that we seek the probability that a permutation on $N - t$ objects has exactly t' fixed points, which would be the additional t' points of agreement between E(K, •) and E(K', •). But a permutation on $N - t$ objects with t' fixed points is equal to the number of ways t' out of $N - t$ objects can be fixed, while the remaining $N - t - t'$ are not fixed. Then using Problem 3.4 we have that

$$\Pr(t' \text{ additional fixed points}) = \binom{N-t}{t'} \times \Pr(\text{no fixed points in } N - t - t' \text{ objects})$$

$$= \frac{1}{(t')!} \times \sum_{k=0}^{N-t-t'} \frac{(-1)^k}{k!}$$

We see that this reduces to the solution to part (a) when $t' = N - t$.

3.4 Let S_{2^n} be the set of permutations on $[0, 1, \ldots, 2^n - 1]$, which is referred to as the symmetric group on 2^n objects, and let $N = 2^n$. For $0 \leq i \leq N$, let A_i be all mappings $\pi \in S_{2^m}$ for which $\pi(i) = i$. It follows that $|A_i| = (N - 1)!$ and $\left|\bigcap_{1 \leq i \leq k} A_i\right| = (N - k)!$. The inclusion-exclusion principle states that

$$\Pr(\text{no fixed points in } \pi) = \frac{1}{N!}\sum_{k=0}^{N}\binom{N}{k} \times (N-k)! \times (-1)^k$$

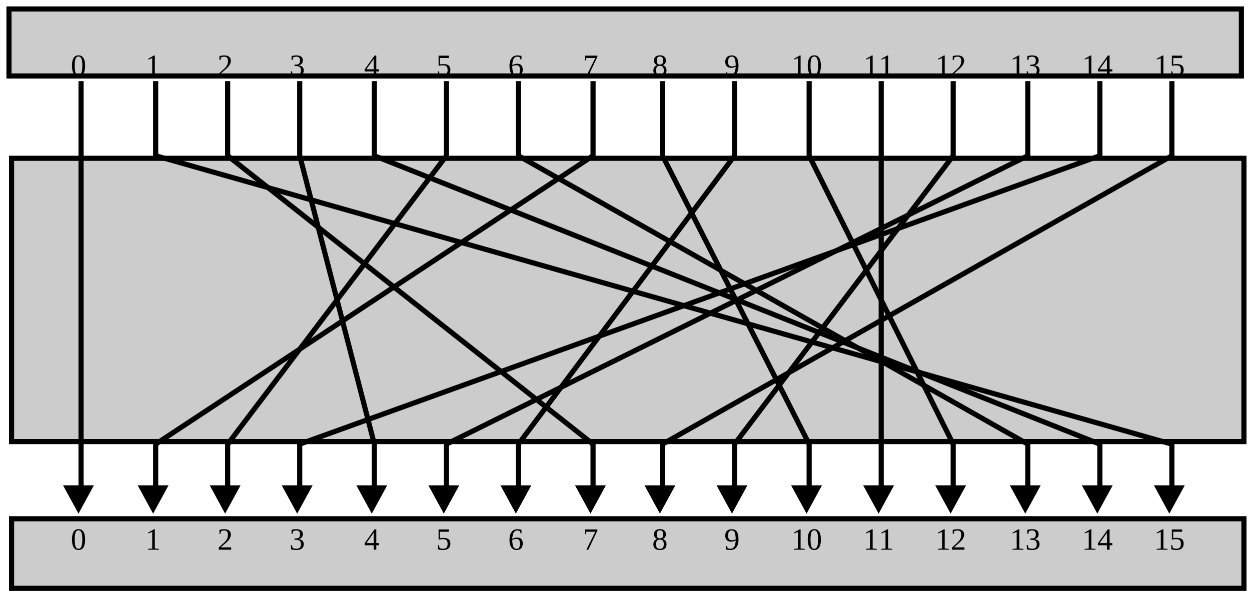
$$= \sum_{k=0}^{N} \frac{(-1)^k}{k!}$$

$$= 1 - 1 + 1/2! - 1/3! + \ldots + (-1)N \times 1/N!$$

$$= e^{-1} + O\left(\frac{1}{N!}\right)$$

Then since $e^{-1} \approx 0.368$, we find that for even small values of N, approximately 37% of permutations contain no fixed points.

3.5



3.6 Main key K = 111…111 (56 bits)
Round keys $K_1 = K_2 = \ldots = K_{16}$ = 1111..111 (48 bits)
Ciphertext C = 1111…111 (64 bits)
Input to the first round of decryption =
$LD_0RD_0 = RE_{16}LE_{16}$ = IP(C) = 1111...111 (64 bits)
$LD_0 = RD_0$ = 1111...111 (32 bits)

Output of the first round of decryption = LD1RD1
LD1 = RD_0 = 1111…111 (32 bits)
Thus, the bits no. 1 and 16 of the output are equal to '1'.

$RD_1 = LD_0 \oplus F(RD_0, K_{16})$
We are looking for bits no. 1 and 16 of RD_1 (33 and 48 of the entire output).

Based on the analysis of the permutation P, bit 1 of $F(RD_0, K_{16})$ comes from the fourth output of the S-box S4, and bit 16 of $F(RD_0, K_{16})$ comes from the second output of the S-box S3. These bits are XOR-ed with 1's from the corresponding positions of LD0.

Inside of the function F,
$E(RD_0) \approx K_{16}$ = 0000…000 (48 bits),
and thus inputs to all eight S-boxes are equal to "000000".

Output from the S-box S4 = "0111", and thus the fourth output is equal to '1',
Output from the S-box S3 = "1010", and thus the second output is equal to '0'.

From here, after the XOR, the bit no. 33 of the first round output is equal to '0', and the bit no. 48 is equal to '1'.

3.7 In the solution given below the following general properties of the XOR function are used:

$$A \oplus 1 = A'$$
$$(A \oplus B)' = A' \oplus B = A \oplus B'$$
$$A' \oplus B' = A \oplus B$$

Where A' = the bitwise complement of A.

a. $F(R_n, K_{n+1}) = \mathbf{1}$

We have
$L_{n+1} = R_n$; $R_{n+1} = L_n \oplus F(R_n, K_{n+1}) = L_n \oplus 1 = L_n'$

Thus
$L_{n+2} = R_{n+1} = L_n'$; $R_{n+2} = L_{n+1} = R_n'$

i.e., after each two rounds we obtain the bit complement of the original input, and every four rounds we obtain back the original input:

$L_{n+4} = L_{n+2}' = L_n$; $R_{n+2} = R_{n+2}' = R_n$

Therefore,
$L_{16} = L_0$; $R_{16} = R_0$

An input to the inverse initial permutation is $R_{16}\, L_{16}$.
Therefore, the transformation computed by the modified DES can be represented as follows:

$C = IP^{-1}(SWAP(IP(M)))$, where SWAP is a permutation exchanging the position of two halves of the input: SWAP(A, B) = (B, A).

This function is linear (and thus also affine). Actually, this is a permutation, the product of three permutations IP, SWAP, and IP^{-1}. This permutation is however different from the identity permutation.

b. $F(R_n, K_{n+1}) = R_n'$

We have

$L_{n+1} = R_n; R_{n+1} = L_n \oplus F(R_n, K_{n+1}) = L_n \oplus R_n'$

$L_{n+2} = R_{n+1} = L_n \oplus R_n'$
$R_{n+2} = L_{n+1} \oplus F(R_{n+1}, K_{n+2}) = R_n \approx (L_n \oplus R_n')' = R_n \oplus L_n \oplus R_n'' = L_n$

$L_{n+3} = R_{n+2} = L_n$
$R_{n+3} = L_{n+2} \oplus F(R_{n+2}, K_{n+3}) = (L_n \approx R_n') \oplus L_n' = R_n' \oplus \mathbf{1} = R_n$

i.e., after each three rounds we come back to the original input.

$L_{15} = L_0; R_{15} = R_0$
and
$L_{16} = R_0$ (1)
$R_{16} = L_0 \oplus R_0'$ (2)

An input to the inverse initial permutation is $R_{16} L_{16}$.
A function described by (1) and (2) is affine, as bitwise complement is affine, and the other transformations are linear.
The transformation computed by the modified DES can be represented as follows:

$C = IP^{-1}(FUN2(IP(M)))$, where $FUN2(A, B) = (A \oplus B', B)$.
This function is affine as a product of three affine functions.

In all cases decryption looks exactly the same as encryption.

3.8 **a.** First, pass the 64-bit input through PC-1 (Table 3.4a) to produce a 56-bit result. Then perform a left circular shift separately on the two 28-bit halves. Finally, pass the 56-bit result through PC-2 (Table 3.4b) to produce the 48-bit K_1.:

in binary notation:	0000 1011 0000 0010 0110 0111
	1001 1011 0100 1001 1010 0101
in hexadecimal notation:	0 B 0 2 6 7 9 B 4 9 A 5

b. L_0, R_0 are derived by passing the 64-plaintext through IP (Table 3.2a):

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111
R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

c. The E table (Table 3.2c) expands R_0 to 48 bits:

$E(R_0)$ = 01110 100001 010101 010101 011110 100001 010101 010101

d. A = 011100 010001 011100 110010 111000 010101 110011 110000

e. $S_1^{00}(1110) = S_1^0(14) =$ 0 (base 10) = 0000 (base 2)
$S_2^{01}(1000) = S_2^1(8) =$ 12 (base 10) = 1100 (base 2)
$S_3^{00}(1110) = S_3^0(14) =$ 2 (base 10) = 0010 (base 2)
$S_4^{10}(1001) = S_4^2(9) =$ 1 (base 10) = 0001 (base 2)
$S_5^{10}(1100) = S_5^2(12) =$ 6 (base 10) = 0110 (base 2)
$S_6^{01}(1010) = S_6^1(10) =$ 13 (base 10) = 1101 (base 2)
$S_7^{11}(1001) = S_7^3(9) =$ 5 (base 10) = 0101 (base 2)
$S_8^{10}(1000) = S_8^2(8) =$ 0 (base 10) = 0000 (base 2)

f. B = 0000 1100 0010 0001 0110 1101 0101 0000

g. Using Table 3.2d, P(B) = 1001 0010 0001 1100 0010 0000 1001 1100

h. R_1 = 0101 1110 0001 1100 1110 1100 0110 0011

i. $L_1 = R_0$. The ciphertext is the concatenation of L_1 and R_1. Source: [MEYE82]

3.9 The reasoning for the Feistel cipher, as shown in Figure 3.6 applies in the case of DES. We only have to show the effect of the IP and IP^{-1} functions. For encryption, the input to the final IP^{-1} is $RE_{16} \| LE_{16}$. The output of that stage is the ciphertext. On decryption, the first step is to take the ciphertext and pass it through IP. Because IP is the inverse of IP^{-1}, the result of this operation is just $RE_{16} \| LE_{16}$, which is equivalent to $LD_0 \| RD_0$. Then, we follow the same reasoning as with the Feistel cipher to reach a point where $LE_0 = RD_{16}$ and $RE_0 = LD_{16}$. Decryption is completed by passing $LD_0 \| RD_0$ through IP^{-1}. Again, because IP is the inverse of IP^{-1}, passing the plaintext through IP as the first step of encryption yields $LD_0 \| RD_0$, thus showing that decryption is the inverse of encryption.

3.10 a. Let us work this from the inside out.

$T_{16}(L_{15} \| R_{15}) = L_{16} \| R_{16}$
$T_{17}(L_{16} \| R_{16}) = R_{16} \| L_{16}$
$IP\,[IP^{-1}\,(R_{16} \| L_{16})] = R_{16} \| L_{16}$
$TD_1(R_{16} \| L_{16}) = R_{15} \| L_{15}$

b. $T_{16}(L_{15} \| R_{15}) = L_{16} \| R_{16}$
$IP\,[IP^{-1}\,(L_{16} \| R_{16})] = L_{16} \| R_{16}$
$TD_1(R_{16} \| L_{16}) = R_{16} \| L_{16} \oplus f(R_{16}, K_{16})$

$\neq L_{15} \| R_{15}$

3.11 PC-1 is essentially the same as IP with every eighth bit eliminated. This would enable a similar type of implementation. Beyond that, there does not appear to be any particular cryptographic significance.

3.12

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3.13 a. The equality in the hint can be shown by listing all 1-bit possibilities:

A	B	$A \oplus B$	$(A \oplus B)'$	$A' \oplus B$
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	0	1	1

We also need the equality $A \oplus B = A' \oplus B'$, which is easily seen to be true. Now, consider the two XOR operations in Figure 3.8. If the plaintext and key for an encryption are complemented, then the inputs to the first XOR are also complemented. The output, then, is the same as for the uncomplemented inputs. Further down, we see that only one of the two inputs to the second XOR is complemented, therefore, the output is the complement of the output that would be generated by uncomplemented inputs.

b. In a chosen plaintext attack, if for chosen plaintext X, the analyst can obtain $Y_1 = E[K, X]$ and $Y_2 = E[K, X']$, then an exhaustive key search requires only 2^{55} rather than 2^{56} encryptions. To see this, note that $(Y_2)' = E[K', X]$. Now, pick a test value of the key T and perform E[T, X]. If the result is Y_1, then we know that T is the correct key. If the result is $(Y_2)'$, then we know that T' is the correct key. If neither result appears, then we have eliminated two possible keys with one encryption.

3.14 The result can be demonstrated by tracing through the way in which the bits are used. An easy, but not necessary, way to see this is to number the 64 bits of the key as follows (read each vertical column of 2 digits as a number):

```
2113355-1025554-0214434-1123334-0012343-2021453-0202435-0110454-
1031975-1176107-2423401-7632789-7452553-0858846-6836043-9495226-
```

The first bit of the key is identified as 21, the second as 10, the third as 13, and so on. The eight bits that are not used in the calculation are unnumbered. The numbers 01 through 28 and 30 through 57 are used. The reason for this assignment is to clarify the way in which the subkeys are chosen. With this assignment, the subkey for the first iteration contains 48 bits, 01 through 24 and 30 through 53, in their natural numerical order. It is easy at this point to see that the first 24 bits of each subkey will always be from the bits designated 01 through 28, and the second 24 bits of each subkey will always be from the bits designated 30 through 57.

3.15 For $1 \le i \le 128$, take $c_i \in \{0, 1\}^{128}$ to be the string containing a 1 in position i and then zeros elsewhere. Obtain the decryption of these 128 ciphertexts. Let m_1, $m_2, \ldots, m_{128}$ be the corresponding plaintexts. Now, given any ciphertext c which does not consist of all zeros, there is a unique nonempty subset of the c_i's which we can XOR together to obtain c. Let $I(c) \subseteq \{1, 2, \ldots, 128\}$ denote this subset. Observe

$$c = \bigoplus_{i \in I(c)} c_i = \bigoplus_{i \in I(c)} E(m_i) = E\left(\bigoplus_{i \in I(c)} m_i\right)$$

Thus, we obtain the plaintext of c by computing $\bigoplus_{i \in I(c)} m_i$. Let **0** be the all-zero string. Note that $\mathbf{0} = \mathbf{0} \oplus \mathbf{0}$. From this we obtain $E(\mathbf{0}) = E(\mathbf{0} \oplus \mathbf{0}) = E(\mathbf{0}) \oplus E(\mathbf{0}) = \mathbf{0}$. Thus, the plaintext of $c = \mathbf{0}$ is $m = \mathbf{0}$. Hence we can decrypt every $c \in \{0, 1\}^{128}$.

3.16 **a.** This adds nothing to the security of the algorithm. There is a one-to-one reversible relationship between the 10-bit key and the output of the P10 function. If we consider the output of the P10 function as a new key, then there are still 2^{10} different unique keys.

b. By the same reasoning as (a), this adds nothing to the security of the algorithm.

3.17 s = wxyz + wxy + wyz + wy + wz + yz + w + x + z
t = wxz + wyz + wz + xz + yz + w + y

3.18 OK

CHAPTER 4
FINITE FIELDS

ANSWERS TO QUESTIONS

4.1 A **group** is a set of elements that is closed under a binary operation and that is associative and that includes an identity element and an inverse element.

4.2 A **ring** is a set of elements that is closed under two binary operations, addition and subtraction, with the following: the addition operation is a group that is commutative; the multiplication operation is associative and is distributive over the addition operation.

4.3 A **field** is a ring in which the multiplication operation is commutative, has no zero divisors, and includes an identity element and an inverse element.

4.4 A nonzero b is a **divisor** of a if $a = mb$ for some m, where a, b, and m are integers. That is, b is a **divisor** of a if there is no remainder on division.

4.5 In modular arithmetic, all arithmetic operations are performed modulo some integer.

4.6 **(1)** Ordinary polynomial arithmetic, using the basic rules of algebra. **(2)** Polynomial arithmetic in which the arithmetic on the coefficients is performed over a finite field; that is, the coefficients are elements of the finite field. **(3)** Polynomial arithmetic in which the coefficients are elements of a finite field, and the polynomials are defined modulo a polynomial $M(x)$ whose highest power is some integer n.

ANSWERS TO PROBLEMS

4.1 **a.** $n!$

b. We can do this by example. Consider the set S_3. We have {3, 2, 1} • {1, 3, 2} = {2, 3, 1}, but {1, 3, 2} • {3, 2, 1} = {3, 1, 2}.

4.2 Here are the addition and multiplication tables

+	0	1	2
0	0	1	2
1	1	2	0

×	0	1	2
0	0	0	0
1	0	1	2

2	2	0	1

2	0	2	1

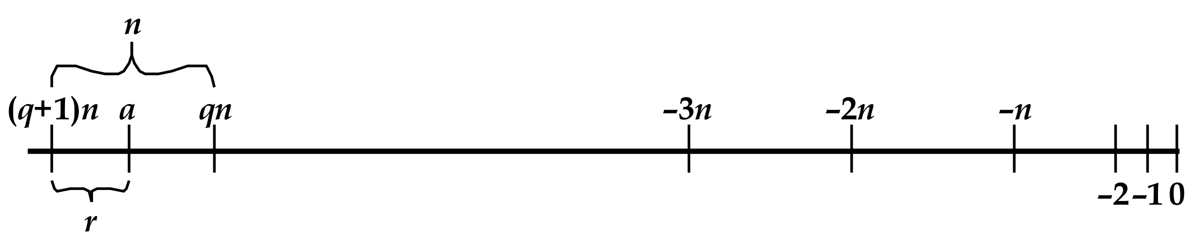
a. Yes. The identity element is 0, and the inverses of 0, 1, 2 are respectively 0, 2, 1.
b. No. The identity element is 1, but 0 has no inverse.

4.3 S is a ring. We show using the axioms in Figure 4.1:

(A1) Closure:	The sum of any two elements in S is also in S.
(A2) Associative:	S is associative under addition, by observation.
(A3) Identity element:	a is the additive identity element for addition.
(A4) Inverse element:	The additive inverses of a and b are b and a, respectively.
(A5) Commutative:	S is commutative under addition, by observation.
(M1) Closure:	The product of any two elements in S is also in S.
(M2) Associative:	S is associative under multiplication, by observation.
(M3) Distributive laws:	S is distributive with respect to the two operations, by observation.

4.4 The equation is the same. For integer $a < 0$, a will either be an integer multiple of n of fall between two consecutive multiples qn and $(q + 1)n$, where $q < 0$. The remainder satisfies the condition $0 \leq r \leq n$.

4.5 In this diagram, q is a negative integer.



4.6 **a.** 2 **b.** 3 **c.** 4 There are other correct answers.

4.7 Section 4.2 defines the relationship: $a = n \times \lfloor a/n \rfloor + (a \bmod n)$. Thus, we can define the mod operator as: $a \bmod n = a - n \times \lfloor a/n \rfloor$.
a. $5 \bmod 3 = 5 - 3 \lfloor 5/3 \rfloor = 2$
b. $5 \bmod -3 = 5 - (-3) \lfloor 5/(-3) \rfloor = -1$
c. $-5 \bmod 3 = -5 - 3 \lfloor (-5)/3 \rfloor = 1$
d. $-5 \bmod -3 = -5 - (-3) \lfloor (-5)/(-3) \rfloor = -2$
This example is from [GRAH94]

4.8 $a = b$

4.9 Recall Figure 4.2 and that any integer a can be written in the form

$$a = qn + r$$

where q is some integer and r one of the numbers

$$0, 1, 2, \ldots, n - 1$$

Using the second definition, no two of the remainders in the above list are congruent (mod n), because the difference between them is less than n and therefore n does not divide that difference. Therefore, two numbers that are not congruent (mod n) must have different remainders. So we conclude that n divides $(a - b)$ if and only if a and b are numbers that have the same remainder when divided by n.

4.10 1, 2, 4, 6, 16, 12

4.11 **a.** This is the definition of congruence as used in Section 4.2.

b. The first two statements mean

$$a - b = nk; \quad b - c = nm$$

so that

$$a - c = (a - b) + (b - c) = n(k + m)$$

4.12 **a.** Let c = a mod n and d = b mod n. Then

$$c = a + kn; \quad d = b + mn; \quad c - d = (a - b) + (k - m)n.$$

Therefore (c – d) = (a – b) mod n

b. Using the definitions of c and d from part (a),

$$cd = ab + n(kb + ma + kmn)$$

Therefore cd = (a × b) mod n

4.13 $1^{-1} = 1$, $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$

4.14 We have $1 \equiv 1 \pmod 9$; $10 \equiv 1 \pmod 9$; $10^2 \equiv 10(10) \equiv 1(1) \equiv 1 \pmod 9$; $10^{n-1} \equiv 1 \pmod 9$. Express N as $a_0 + a_1 10^1 + \ldots + a_{n-1}10^{n-1}$. Then $N \equiv a_0 + a_1 + \ldots + a_{n-1} \pmod 9$.

4.15 **a.** gcd(24140, 16762) = gcd(16762, 7378) = gcd(7378, 2006) = gcd(2006, 1360) = gcd(1360, 646) = gcd (646, 68) = gcd(68, 34) = gcd(34, 0) = 34

b. 35

4.16 **a.** We want to show that m > 2r. This is equivalent to qn + r > 2r, which is equivalent to qn > r. Since n > r, we must have qn > r.

b. If you study the pseudocode for Euclid's algorithm in the text, you can see that the relationship defined by Euclid's algorithm can be expressed as

$$A_i = q_{i+1}A_{i+1} + A_{i+2}$$

The relationship $A_{i+2} < A_i/2$ follows immediately from (a).

c. From (b), we see that $A_3 < 2^{-1}A_1$, that $A_5 < 2^{-1}A_3 < 2^{-2}A_5$, and in general that $A_{2j+1} < 2^{-j}A_1$ for all integers j such that $1 < 2j + 1 \leq k + 2$, where k is the number of steps in the algorithm. If k is odd, we take $j = (k + 1)/2$ to obtain $N > (k + 1)/2$, and if k is even, we take $j = k/2$ to obtain $N > k/2$. In either case $k < 2N$.

4.17 a. **Euclid:** gcd(2152, 764) = gcd(764, 624) = gcd(624, 140) = gcd(140, 64) = gcd(64, 12) = gcd(12, 4) = gcd(4, 0) = 4

Stein: $A_1 = 2152$, $B_1 = 764$, $C_1 = 1$; $A_2 = 1076$, $B_2 = 382$, $C_2 = 2$; $A_3 = 538$, $B_3 = 191$, $C_3 = 4$; $A_4 = 269$, $B_4 = 191$, $C_4 = 4$; $A_5 = 78$, $B_5 = 191$, $C_5 = 4$; $A_5 = 39$, $B_5 = 191$, $C_5 = 4$; $A_6 = 152$, $B_6 = 39$, $C_6 = 4$; $A_7 = 76$, $B_7 = 39$, $C_7 = 4$; $A_8 = 38$, $B_8 = 39$, $C_8 = 4$; $A_9 = 19$, $B_9 = 39$, $C_9 = 4$; $A_{10} = 20$, $B_{10} = 19$, $C_{10} = 4$; $A_{11} = 10$, $B_{11} = 19$, $C_{11} = 4$; $A_{12} = 5$, $B_{12} = 19$, $C_{12} = 4$; $A_{13} = 14$, $B_{13} = 5$, $C_{13} = 4$; $A_{14} = 7$, $B_{14} = 5$, $C_{14} = 4$; $A_{15} = 2$, $B_{15} = 5$, $C_{15} = 4$; $A_{16} = 1$, $B_{16} = 5$, $C_{16} = 4$; $A_{17} = 4$, $B_{17} = 1$, $C_{17} = 4$; $A_{18} = 2$, $B_{18} = 1$, $C_{18} = 4$; $A_{19} = 1$, $B_{19} = 1$, $C_{19} = 4$; gcd(2152, 764) = $1 \times 4 = 4$

b. Euclid's algorithm requires a "long division" at each step whereas the Stein algorithm only requires division by 2, which is a simple operation in binary arithmetic.

4.18 a. If A_n and B_n are both even, then $2 \times \gcd(A_{n+1}, B_{n+1}) = \gcd(A_n, B_n)$. But $C_{n+1} = 2C_n$, and therefore the relationship holds.

If one of A_n and B_n is even and one is odd, then dividing the even number does not change the gcd. Therefore, $\gcd(A_{n+1}, B_{n+1}) = \gcd(A_n, B_n)$. But $C_{n+1} = C_n$, and therefore the relationship holds.

If both A_n and B_n are odd, we can use the following reasoning based on the rules of modular arithmetic. Let $D = \gcd(A_n, B_n)$. Then D divides $|A_n - B_n|$ and D divides $\min(A_n, B_n)$. Therefore, $\gcd(A_{n+1}, B_{n+1}) = \gcd(A_n, B_n)$. But $C_{n+1} = C_n$, and therefore the relationship holds.

b. If at least one of A_n and B_n is even, then at least one division by 2 occurs to produce A_{n+1} and B_{n+1}. Therefore, the relationship is easily seen to hold. Suppose that both A_n and B_n are odd; then A_{n+1} is even; in that case the relationship obviously holds.

c. By the result of (b), every 2 iterations reduces the AB product by a factor of 2. The AB product starts out at $< 2^{2N}$. There are at most $\log(2^{2N}) = 2N$ pairs of iterations, or at most 4N iterations.

d. At the very beginning, we have $A_1 = A$, $B_1 = B$, and $C_1 = 1$. Therefore $C_1 \times \gcd(A_1, B_1) = \gcd(A, B)$. Then, by (a), $C_2 \times \gcd(A_2, B_2) = C_1 \times \gcd(A_1, B_1) = \gcd(A, B)$. Generalizing, $C_n \times \gcd(A_n, B_n) = \gcd(A, B)$. The algorithm stops when $A_n = B_n$. But, for $A_n = B_n$, $\gcd(A_n, B_n) = A_n$. Therefore, $C_n \times \gcd(A_n, B_n) = C_n \times A_n = \gcd(A, B)$.

4.19 **a.** 3239

b. gcd(40902, 24240) = 34 ≠ 1, so there is no multiplicative inverse.

c. 550

4.20

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

w	$-w$	w^{-1}
0	0	–
1	4	1
2	3	3
3	2	2
4	1	4

4.21 Let S be the set of polynomials whose coefficients form a field F. Recall that addition is defined as follows: For

$$f(x) = \sum_{i=0}^{n} a_i x^i; \quad g(x) = \sum_{i=0}^{m} b_i x^i; \qquad n \geq m$$

then addition is defined as:

$$f(x) + g(x) = \sum_{i=0}^{m} (a_i + b_i) x^i + \sum_{i=m+1}^{n} a_i x^i$$

Using the axioms in Figure 4.1, we now examine the addition operation:

(A1) Closure: The sum of any two elements in S is also in S. This is so because the sum of any two coefficients is also a valid coefficient, because F is a field.

(A2) Associative: S is associative under addition. This is so because coefficient addition is associative.

(A3) Identity element: 0 is the additive identity element for addition.

(A4) Inverse element: The additive inverse of a polynomial f(x) a polynomial with the coefficients $-a_i$.

(A5) Commutative: S is commutative under addition. This is so because coefficient addition is commutative.

Multiplication is defined as follows:

$$f(x) \times g(x) = \sum_{i=0}^{n+m} c_i x^i$$

where

$$c_k = a_0 b_k + a_1 b_{k-1} + \cdots + a_{k-1} b_1 + a_k b_0$$

In the last formula, we treat a_i as zero for $i > n$ and b_i as zero for $i > m$.

(M1) Closure: The product of any two elements in S is also in S. This is so because the product of any two coefficients is also a valid coefficient, because F is a field.

(M2) Associative: S is associative under multiplication. This is so because coefficient multiplication is associative.

(M3) Distributive laws: S is distributive with respect to the two operations, by the field properties of the coefficients.

4.22 a. True. To see, this consider the equation for c_k, above, for k = n + m, where f(x) and g(x) are monic. The only nonzero term on the right of equation is $a_n b_m$, which has the value 1.

b. True. We have $c_{n+m} = a_n b_m \neq 0$.

c. True when $m \neq n$; in that case the highest degree coefficient is of degree max[m,n]. But false in general when m = n, because the highest-degree coefficients might cancel (be additive inverses).

4.23 a. $9x^2 + 7x + 7$

b. $5x^3 + 7x^2 + 2x + 6$

4.24 a. Reducible: $(x + 1)(x^2 + x + 1)$

b. Irreducible. If you could factor this polynomial, one factor would be either x or (x + 1), which would give you a root of x = 0 or x = 1 respectively. By substitution of 0 and 1 into this polynomial, it clearly has no roots.

c. Reducible: $(x + 1)^4$

4.25 a. 1

b. 1

c. x + 1

d. x + 78 Source: [KOBL94]

4.26 Polynomial Arithmetic Modulo $(x^2 + x + 1)$:

	+	000 0	001 1	010 x	011 $x + 1$
000	0	0	1	x	$x + 1$
001	1	1	0	$x + 1$	x
010	x	x	$x + 1$	0	1
011	$x + 1$	$x + 1$	x	1	0

	×	000 0	001 1	010 x	011 $x + 1$
000	0	0	0	0	0
001	1	0	1	x	$x + 1$
010	x	0	x	$x + 1$	1
011	$x + 1$	0	$x + 1$	1	x

4.27 $x^2 + 1$

4.28 Generator for $GF(2^4)$ using $x^4 + x + 1$

Power Representation	**Polynomial Representation**	**Binary Representation**	**Decimal (Hex) Representation**
0	0	0000	0
g^0 (= g^{15})	1	0001	1
g^1	g	0010	2
g^2	g^2	0100	4
g^3	g^3	1000	8
g^4	$g + 1$	0011	3
g^5	$g^2 + g$	0110	6
g^6	$g^3 + g^2$	1100	12
g^7	$g^3 + g + 1$	1011	11
g^8	$g^2 + 1$	0101	5
g^9	$g^3 + g$	1010	10
g^{10}	$g^2 + g + 1$	0111	7
g^{11}	$g^3 + g^2 + g$	1110	14
g^{12}	$g^3 + g^2 + g + 1$	1111	15

g^{13}	$g^3 + g^2 + 1$	1101	13
g^{14}	$g^3 + 1$	1001	9

CHAPTER 5
ADVANCED ENCRYPTION STANDARD

ANSWERS TO QUESTIONS

5.1 **Security:** Actual security; randomness; soundness, other security factors.
Cost: Licensing requirements; computational efficiency; memory requirements.
Algorithm and Implementation Characteristics: Flexibility; hardware and software suitability; simplicity.

5.2 General security; software implementations; restricted-space environments; hardware implementations; attacks on implementations; encryption vs. decryption; key agility; other versatility and flexibility; potential for instruction-level parallelism.

5.3 The basic idea behind power analysis is the observation that the power consumed by a smart card at any particular time during the cryptographic operation is related to the instruction being executed and to the data being processed.

5.4 Rijndael allows for block lengths of 128, 192, or 256 bits. AES allows only a block length of 128 bits.

5.5 The State array holds the intermediate results on the 128-bit block at each stage in the processing.

5.6 **1.** Initialize the S-box with the byte values in ascending sequence row by row. The first row contains {00}, {01}, {02}, etc., the second row contains {10}, {11}, etc., and so on. Thus, the value of the byte at row x, column y is $\{xy\}$.
2. Map each byte in the S-box to its multiplicative inverse in the finite field $GF(2^8)$; the value {00} is mapped to itself.
3. Consider that each byte in the S-box consists of 8 bits labeled (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0). Apply the following transformation to each bit of each byte in the S-box:

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$$

where c_i is the ith bit of byte c with the value {63}; that is, $(c_7c_6c_5c_4c_3c_2c_1c_0)$ = (01100011). The prime (') indicates that the variable is to be updated by the value on the right.

5.7 Each individual byte of **State** is mapped into a new byte in the following way: The leftmost 4 bits of the byte are used as a row value and the rightmost 4 bits are used as a column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value.

5.8 The first row of **State** is not altered. For the second row, a 1-byte circular left shift is performed. For the third row, a 2-byte circular left shift is performed. For the third row, a 3-byte circular left shift is performed.

5.9 12 bytes.

5.10 MixColumns operates on each column individually. Each byte of a column is mapped into a new value that is a function of all four bytes in that column.

5.11 The 128 bits of **State** are bitwise XORed with the 128 bits of the round key.

5.12 The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (156 bytes). The expansion is defined by the pseudocode in Section 5.2.

5.13 SubBytes operates on State, with each byte mapped into a new byte using the S-box. SubWord operates on an input word, with each byte mapped into a new byte using the S-box.

5.14 ShiftRows is described in the answer to Question 5.8. RotWord performs a one-byte circular left shift on a word; thus it is equivalent to the operation of ShiftRows on the second row of State.

5.15 For the AES decryption algorithm, the sequence of transformations for decryption differs from that for encryption, although the form of the key schedules for encryption and decryption is the same. The equivalent version has the same sequence of transformations as the encryption algorithm (with transformations replaced by their inverses). To achieve this equivalence, a change in key schedule is needed.

Answers to Problems

5.1 We want to show that d(x) = a(x) x b(x) mod (x^4 + 1) = 1. Substituting into Equation (5.12) in Appendix 5A, we have:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 0E \\ 09 \\ 0D \\ 0B \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

But this is the same set of equations discussed in the subsection on the MixColumn transformation:

$$\begin{aligned} (\{0E\} \bullet \{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\} \bullet \{03\}) &= \{01\} \\ (\{09\} \bullet \{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\} \bullet \{03\}) &= \{00\} \\ (\{0D\} \bullet \{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\} \bullet \{03\}) &= \{00\} \\ (\{0B\} \bullet \{02\}) \oplus \{0D\} \oplus \{09\} \oplus (\{0E\} \bullet \{03\}) &= \{00\} \end{aligned}$$

The first equation is verified in the text. For the second equation, we have {09} • {02} = 00010010; and {0D} • {03} = {0D} ⊕ ({0D} • {02}) = 00001101 ⊕ 00011010 = 00010111. Then

$$\begin{array}{lcl} \{09\} \bullet \{02\} & = & 00010010 \\ \{0E\} & = & 00001110 \\ \{0B\} & = & 00001011 \\ \{0D\} \bullet \{03\} & = & \underline{00010111} \\ & & 00000000 \end{array}$$

For the third equation, we have {0D} • {02} = 00011010; and {0B} • {03} = {0B} ⊕ ({0B} • {02}) = 00001011 ⊕ 00010110 = 00011101. Then

$$\begin{array}{lcl} \{0D\} \bullet \{02\} & = & 00011010 \\ \{09\} & = & 00001001 \\ \{0E\} & = & 00001110 \\ \{0B\} \bullet \{03\} & = & \underline{00011101} \\ & & 00000000 \end{array}$$

For the fourth equation, we have {0B} • {02} = 00010110; and {0E} • {03} = {0E} ⊕ ({0E} • {02}) = 00001110 ⊕ 00011100 = 00010010. Then

$$\begin{array}{lcl} \{0B\} \bullet \{02\} & = & 00010110 \\ \{0D\} & = & 00001101 \\ \{09\} & = & 00001001 \\ \{0E\} \bullet \{03\} & = & \underline{00010010} \\ & & 00000000 \end{array}$$

5.2 **a.** {01}

b. We need to show that the transformation defined by Equation 5.2, when applied to $\{01\}^{-1}$, produces the correct entry in the S-box. We have

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The result is {7C}, which is the same as the value for {01} in the S-box (Table 5.4a).

5.3 w(0) = {00 00 00 00}; w(1) = {00 00 00 00}; w(2) = {00 00 00 00}; w(3) = {00 00 00 00}; w(4) = {62 63 63 63}; w(5) = {62 63 63 63}; w(6) = {62 63 63 63}; w(7) = {62 63 63 63}

5.4

00	04	08	0C
01	05	09	0D
02	06	0A	0E
03	07	0B	0F

a

01	05	09	0D
00	04	08	0C
03	07	0B	0F
02	06	0A	0E

b

7C	6B	01	D7
63	F2	30	FE
7B	C5	2B	76
77	6F	67	AB

c

7C	6B	01	D7
F2	30	FE	63
2B	76	7B	C5
AB	77	6F	67

d

75	87	0F	A2
55	E6	04	22
3E	2E	B8	8C
10	15	58	0A

e

5.5 It is easy to see that $x^4 \bmod (x^4 + 1) = 1$. This is so because we can write:

$$x^4 = [1 \times (x^4 + 1)] + 1$$

Recall that the addition operation is XOR. Then,

$$x^8 \bmod (x^4 + 1) = [x^4 \bmod (x^4 + 1)] \times [x^4 \bmod (x^4 + 1)] = 1 \times 1 = 1$$

So, for any positive integer a, $x^{4a} \bmod (x^4 + 1) = 1$. Now consider any integer i of the form $i = 4a + (i \bmod 4)$. Then,

$$x^i \bmod (x^4 + 1) = [(x^{4a}) \times (x^{i \bmod 4})] \bmod (x^4 + 1)$$
$$= [x^{4a} \bmod (x^4 + 1)] \times [x^{i \bmod 4} \bmod (x^4 + 1)] = x^{i \bmod 4}$$

The same result can be demonstrated using long division.

5.6 **a.** AddRoundKey

b. The MixColumn step, because this is where the different bytes interact with each other.

c. The ByteSub step, because it contributes nonlinearity to AES.

d. The ShiftRow step, because it permutes the bytes.

e. There is no wholesale swapping of rows or columns. AES does not require this step because: The MixColumn step causes every byte in a column to alter every other byte in the column, so there is not need to swap rows; The ShiftRow step moves bytes from one column to another, so there is no need to swap columns

Source: These observations were made by John Savard

5.7 The primary issue is to assure that multiplications take a constant amount of time, independent of the value of the argument. This can be done by adding no-operation cycles as needed to make the times uniform.

5.8

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{vmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{vmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

5.9 Input = 67 89 AB CD.

$$\text{Output} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 67 \\ 89 \\ AB \\ CD \end{bmatrix} = \begin{bmatrix} 67\cdot 2 + 89\cdot 3 + AB + CD \\ 67 + 89\cdot 2 + AB\cdot 3 + CD \\ 67 + 89 + AB\cdot 2 + CD\cdot 3 \\ 67\cdot 3 + 89 + AB + CD\cdot 2 \end{bmatrix} = \begin{bmatrix} CE + 80 + AB + CD \\ 67 + 09 + E6 + CD \\ 67 + 89 + 4D + 4C \\ A9 + 89 + AB + 81 \end{bmatrix} = \begin{bmatrix} 28 \\ 45 \\ EF \\ 0A \end{bmatrix}$$

Verification with the Inverse Mix Column transformation gives

$$\text{Input}'' = \begin{bmatrix} E & B & D & 9 \\ 9 & E & B & D \\ D & 9 & E & B \\ B & D & 9 & E \end{bmatrix} \begin{bmatrix} 28 \\ 45 \\ EF \\ 0A \end{bmatrix} = \begin{bmatrix} 28\cdot E + 45\cdot B + EF\cdot D + 0A\cdot 9 \\ 28\cdot 9 + 45\cdot E + EF\cdot B + 0A\cdot D \\ 28\cdot D + 45\cdot 9 + EF\cdot E + 0A\cdot B \\ 28\cdot B + 45\cdot D + EF\cdot 9 + 0A\cdot E \end{bmatrix} = \begin{bmatrix} AB + D1 + 47 + 5A \\ 73 + 9B + 13 + 72 \\ D3 + 5B + 6D + 4E \\ 23 + 54 + D6 + 6C \end{bmatrix} = \begin{bmatrix} 67 \\ 89 \\ AB \\ CD \end{bmatrix}$$

After changing one bit in the input,
Input′ = 77 89 AB CD,
and the corresponding output

$$\text{Output}' = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 77 \\ 89 \\ AB \\ CD \end{bmatrix} = \begin{bmatrix} 77\cdot 2 + 89\cdot 3 + AB + CD \\ 77 + 89\cdot 2 + AB\cdot 3 + CD \\ 77 + 89 + AB\cdot 2 + CD\cdot 3 \\ 77\cdot 3 + 89 + AB + CD\cdot 2 \end{bmatrix} = \begin{bmatrix} EE + 80 + AB + CD \\ 77 + 89 + E6 + CD \\ 77 + 89 + 4D + 4C \\ C7 + 89 + AB + 81 \end{bmatrix} = \begin{bmatrix} 08 \\ 55 \\ FF \\ 3A \end{bmatrix}$$

The number of bits that changed in the output as a result of a single-bit change in the input is 5.

5.10 Key expansion:

W0 = 1010 0111 W1 = 0011 1011 W2 = 0001 1100 W3 = 0010 0111
W4 = 0111 0110 W5 = 0101 0001

Round 0:

After Add round key: 1100 1000 0101 0000

Round 1:

After Substitute nibbles: 1100 0110 0001 1001
After Shift rows: 1100 1001 0001 0110
After Mix columns: 1110 1100 1010 0010
After Add round key: 1110 1100 1010 0010

Round 2:

After Substitute nibbles: 1111 0000 1000 0101
After Shift rows: 0111 0001 0110 1001
After Add round key: 0000 0111 0011 1000

5.11 $\begin{bmatrix} x^3+1 & x \\ x & x^3+1 \end{bmatrix}\begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

To get the above result, observe that $(x^5 + x^2 + x) \bmod (x^4 + x + 1) = 0$

5.12 The decryption process should be the reverse of the encryption process.

CHAPTER 6
MORE ON SYMMETRIC CIPHERS

ANSWERS TO QUESTIONS

6.1 With triple encryption, a plaintext block is encrypted by passing it through an encryption algorithm; the result is then passed through the same encryption algorithm again; the result of the second encryption is passed through the same encryption algorithm a third time. Typically, the second stage uses the decryption algorithm rather than the encryption algorithm.

6.2 This is an attack used against a double encryption algorithm and requires a known (plaintext, ciphertext) pair. In essence, the plaintext is encrypted to produce an intermediate value in the double encryption, and the ciphertext is decrypted to produce an intermediation value in the double encryption. Table lookup techniques can be used in such a way to dramatically improve on a brute-force try of all pairs of keys.

6.3 Triple encryption can be used with three distinct keys for the three stages; alternatively, the same key can be used for the first and third stage.

6.4 There is no cryptographic significance to the use of decryption for the second stage. Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES by repeating the key.

6.5 **1.** The encryption sequence should have a large period. **2.**The keystream should approximate the properties of a true random number stream as close as possible. **3.** To guard against brute-force attacks, the key needs to be sufficiently long. The same considerations as apply for block ciphers are valid here. Thus, with current technology, a key length of at least 128 bits is desirable.

6.6 If two plaintexts are encrypted with the same key using a stream cipher, then cryptanalysis is often quite simple. If the two ciphertext streams are XORed together, the result is the XOR of the original plaintexts. If the plaintexts are text strings, credit card numbers, or other byte streams with known properties, then cryptanalysis may be successful.

6.7 The actual encryption involves only the XOR operation. Key stream generation involves the modulo operation and byte swapping.

6.8 In some modes, the plaintext does not pass through the encryption function, but is XORed with the output of the encryption function. The math works out that for decryption in these cases, the encryption function must also be used.

Answers to Problems

6.1 **a.** If the IVs are kept secret, the 3-loop case has more bits to be determined and is therefore more secure than 1-loop for brute force attacks.

b. For software implementations, the performance is equivalent for most measurements. One-loop has two fewer XORs per block. three-loop might benefit from the ability to do a large set of blocks with a single key before switching. The performance difference from choice of mode can be expected to be smaller than the differences induced by normal variation in programming style.

For hardware implementations, three-loop is three times faster than one-loop, because of pipelining. That is: Let P_i be the stream of input plaintext blocks, X_i the output of the first DES, Y_i the output of the second DES and C_i the output of the final DES and therefore the whole system's ciphertext.

In the 1-loop case, we have:

X_i = DES(XOR(P_i, C_{i-1}))
Y_i = DES(X_i)
C_i = DES(Y_i)

[where C_0 is the single IV]

If P_1 is presented at t=0 (where time is measured in units of DES operations), X_1 will be available at t=1, Y_1 at t=2 and C_1 at t=3. At t=1, the first DES is free to do more work, but that work will be:

X_2 = DES(XOR(P_2, C_1))

but C_1 is not available until t=3, therefore X_2 can not be available until t=4, Y_2 at t=5 and C_2 at t=6.

In the 3-loop case, we have:

X_i = DES(XOR(P_i, X_{i-1}))
Y_i = DES(XOR(X_i, Y_{i-1}}))

C_i = DES(XOR(Y_i, C_{i-1}))

[where X_0, Y_0 and C_0 are three independent IVs]

If P_1 is presented at t=0, X1 is available at t=1. Both X_2 and Y_1 are available at t=4. X_3, Y_2 and C_1 are available at t=3. X_4, Y_3 and C_2 are available at t=4. Therefore, a new ciphertext block is produced every 1 tick, as opposed to every 3 ticks in the single-loop case. This gives the three-loop construct a throughput three times greater than the one-loop construct.

6.2 Instead of CBC [CBC (CBC (X))], use ECB [CBC (CBC (X))]. The final IV was not needed for security. The lack of feedback loop prevents the chosen-ciphertext differential cryptanalysis attack. The extra IVs still become part of a key to be determined during any known plaintext attack.

6.3 The Merkle-Hellman attack finds the desired two keys K_1 and K_2 by finding the plaintext-ciphertext pair such that intermediate value A is 0. The first step is to create a list of all of the plaintexts that could give A = 0:

$$P_i = D[i, 0] \qquad \text{for } i = 0.\ 1.\ \ldots,\ 2^{56} - 1$$

Then, use each P_i as a chosen plaintext and obtain the corresponding ciphertexts C_i:

$$C_i = E[i, P_i] \qquad \text{for } i = 0.\ 1.\ \ldots,\ 2^{56} - 1$$

The next step is to calculate the intermediate value B_i for each C_i using $K_3 = K_1 = i$.

$$B_i = D[i, C_i] \qquad \text{for } i = 0.\ 1.\ \ldots,\ 2^{56} - 1$$

A table of triples of the following form is constructed: (P_i or B_i, *i*, *flag*), where *flag* indicates either a P-type or B-type triple. Note that the 256 values P_i are also potentially intermediate values B. All P_i and B_i values are placed in the table, and the table is sorted on the first entry in each triple, and then search to find consecutive P and B values such that $B_i = P_j$. For each such equality, *i*, *j* is a candidate for the desired pair of keys K_1 and K_4. Each candidate pair of keys is tested on a few other plaintext-ciphertext pairs to filter out false alarms.

6.4 **a.** No. For example, suppose C_1 is corrupted. The output block P_3 depends only on the input blocks C_2 and C_3.

b. An error in P_1 affects C_1. But since C_1 is input to the calculation of C_2, C_2 is affected. This effect carries through indefinitely, so that all ciphertext blocks are affected. However, at the receiving end, the decryption algorithm restores the correct plaintext for blocks except the one in error. You can show this by writing out the equations for the decryption. Therefore, the error only effects the corresponding decrypted plaintext block.

6.5 Nine plaintext characters are affected. The plaintext character corresponding to the ciphertext character is obviously altered. In addition, the altered ciphertext character enters the shift register and is not removed until the next eight characters are processed.

6.6

Mode	Encrypt	Decrypt
ECB	$C_j = E(K, P_j) \quad j = 1, \ldots, N$	$P_j = D(K, C_j) \quad j = 1, \ldots, N$
CBC	$C_1 = E(K, [P_1 \oplus IV])$ $C_j = E(K, [P_j \oplus C_{j-1}]) \quad j = 2, \ldots, N$	$P_1 = D(K, C_1) \oplus IV$ $P_j = D(K, C_j) \oplus C_{j-1} \quad j = 2, \ldots, N$
CFB	$C_1 = P_1 \oplus S_s(E[K, IV])$ $C_j = P_j \oplus S_s(E[K, C_{j-1}])$	$P_1 = C_1 \oplus S_s(E[K, IV])$ $P_j = C_j \oplus S_s(E[K, C_{j-1}])$
OFB	$C_1 = P_1 \oplus S_s(E[K, IV])$ $C_j = P_j \oplus S_s(E(K, [C_{j-1} \oplus P_{j-1}]))$	$P_1 = C_1 \oplus S_s(E[K, IV])$ $P_j = C_j \oplus S_s(E(K, [C_{j-1} \oplus P_{j-1}]))$
CTR	$C_j = P_j \oplus E[K, Counter + j - 1]$	$P_j = C_j \oplus E[K, Counter + j - 1]$

6.7 After decryption, the last byte of the last block is used to determine the amount of padding that must be stripped off. Therefore there must be at least one byte of padding.

6.8 **a.** Assume that the last block of plaintext is only L bytes long, where $L < 2w/8$. The encryption sequence is as follows (The description in RFC 2040 has an error; the description here is correct.):

1. Encrypt the first $(N - 2)$ blocks using the traditional CBC technique.
2. XOR P_{N-1} with the previous ciphertext block C_{N-2} to create Y_{N-1}.
3. Encrypt Y_{N-1} to create E_{N-1}.
4. Select the first L bytes of E_{N-1} to create C_N.
5. Pad P_N with zeros at the end and exclusive-OR with E_{N-1} to create Y_N.
6. Encrypt Y_N to create C_{N-1}.

The last two blocks of the ciphertext are C_{N-1} and C_N.

b. $P_{N-1} = C_{N-2} \oplus D(K, [C_N \| X])$
$P_N \| X = (C_N \| 00\ldots0) \oplus D(K, [C_{N-1}])$
P_N = left-hand portion of $(P_N \| X)$
where $\|$ is the concatenation function

6.9 **a.** Assume that the last block (P_N) has j bits. After encrypting the last full block (P_{N-1}), encrypt the ciphertext (C_{N-1}) again, select the leftmost j bits of the encrypted ciphertext, and XOR that with the short block to generate the output ciphertext.

b. While an attacker cannot recover the last plaintext block, he can change it systematically by changing individual bits in the ciphertext. If the last few bits of the plaintext contain essential information, this is a weakness.

6.10 Use a key of length 255 bytes. The first two bytes are zero; that is K[0] = K[1] = 0. Thereafter, we have: K[2] = 255; K[3] = 254; … K[255]= 2.

6.11 **a.** Simply store i, j, and S, which requires 8 + 8 + (256 × 8) = 2064 bits

b. The number of states is $[256! \times 256^2] \approx 2^{1700}$. Therefore, 1700 bits are required.

6.12 **a.** By taking the first 80 bits of $v \| c$, we obtain the initialization vector, v. Since v, c, k are known, the message can be recovered (i.e., decrypted) by computing $\mathrm{RC4}(v \| k) \oplus c$.

b. If the adversary observes that $v_i = v_j$ for distinct i, j then he/she knows that the same key stream was used to encrypt both m_i and m_j. In this case, the messages m_i and m_j may be vulnerable to the type of cryptanalysis carried out in part (a).

c. Since the key is fixed, the key stream varies with the choice of the 80-bit v, which is selected randomly. Thus, after approximately $\sqrt{\frac{\pi}{2} 2^{80}} \approx 2^{40}$ messages are sent, we expect the same v, and hence the same key stream, to be used more than once.

d. The key k should be changed sometime before 2^{40} messages are sent.

CHAPTER 7
CONFIDENTIALITY USING SYMMETRIC ENCRYPTION

ANSWERS TO QUESTIONS

7.1 LAN, dial-in communications server, Internet, wiring closet.

7.2 With **link encryption**, each vulnerable communications link is equipped on both ends with an encryption device. With **end-to-end encryption**, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data; the data in encrypted form are then transmitted unaltered across the network to the destination terminal or host.

7.3 Identities of partners. How frequently the partners are communicating. Message pattern, message length, or quantity of messages that suggest important information is being exchanged. The events that correlate with special conversations between particular partners

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

7.5 For two parties A and B, key distribution can be achieved in a number of ways, as follows:

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

7.6 A **session key** is a temporary encryption key used between two principals. A **master key** is a long-lasting key that is used between a key distribution center and a principal for the purpose of encoding the transmission of session keys. Typically, the master keys are distributed by noncryptographic means.

7.7 A nonce is a value that is used only once, such as a timestamp, a counter, or a random number; the minimum requirement is that it differs with each transaction.

7.8 A key distribution center is a system that is authorized to transmit temporary session keys to principals. Each session key is transmitted in encrypted form, using a master key that the key distribution center shares with the target principal.

7.9 Statistical randomness refers to a property of a sequence of numbers or letters, such that the sequence appears random and passes certain statistical tests that indicate that the sequence has the properties of randomness. If a statistically random sequence is generated by an algorithm, then the sequence is predictable by anyone knowing the algorithm and the starting point of the sequence. An unpredictable sequence is one in which knowledge of the sequence generation method is insufficient to determine the sequence.

ANSWERS TO PROBLEMS

7.1 **a.** Mail-bagging economizes on data transmission time and costs. It also reduces the amount of temporary storage that each intermediate system must have available to buffer messages in its possession. These factors can be very significant in electronic mail systems that process a large number of messages. Routing decisions may keep mail-bagging in mind. Implementing mail-bagging adds slightly to the complexity of the forwarding protocol.

b. If a standardized scheme such as PGP or S/MIME is used, then the message is encrypted and both systems should be equally secure.

7.2 **1.** The timing of message transmissions may be varied, with the amount of time between messages serving as the covert channel.

2. A message could include a name of a file; the length of the filename could function as a covert channel.

3. A message could report on the amount of available storage space; the value could function as a covert channel.

7.3 **a.** A sends a connection request to B, with an event marker or nonce (Na) encrypted with the key that A shares with the KDC. If B is prepared to accept the connection, it sends a request to the KDC for a session key, including A's encrypted nonce plus a nonce generated by B (Nb) and encrypted with the key that B shares with the KDC. The KDC returns two encrypted blocks to B. One block is intended for B and includes the session key, A's identifier, and B's nonce. A similar block is prepared for A and passed from the KDC to B and then to A. A and B have now securely obtained the session key and, because of the nonces, are assured that the other is authentic.

b. The proposed scheme appears to provide the same degree of security as that of Figure 7.9. One advantage of the proposed scheme is that the, in the event that B rejects a connection, the overhead of an interaction with the KDC is avoided.

7.4 **i)** Sending to the server the source name A, the destination name Z (his own), and E(K_a, R), as if A wanted to send him the same message encrypted under the same key R as A did it with B

ii) The server will respond by sending E(K_z, R) to A and Z will intercept that

iii) because Z knows his key K_z, he can decrypt E(K_z, R), thus getting his hands on R that can be used to decrypt E(R, M) and obtain M.

7.5 We give the result for $a = 3$:

1, 3, 9, 27, 19, 26, 16, 17, 20, 29, 25, 13, 8, 24, 10, 30, 28, 22, 4, 12, 5, 15, 14, 11, 2, 6, 18, 23, 7, 21, 1

7.6 **a.** Maximum period is $2^{4-2} = 4$
b. a must be 5 or 11
c. The seed must be odd

7.7 When $m = 2^k$, the right-hand digits of X_n are much less random than the left-hand digits. See [KNUT98], page 13 for a discussion.

7.8 Let us start with an initial seed of 1. The first generator yields the sequence:

1, 6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11, 1, . . .

The second generator yields the sequence:

1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1, . . .

Because of the patterns evident in the second half of the latter sequence, most people would consider it to be less random than the first sequence.

7.9 Many packages make use of a linear congruential generator with $m = 2^k$. As discussed in the answer to Problem 5.6, this leads to results in which the right-hand digits are much less random than the left-hand digits. Now, if we use a linear congruential generator of the following form:

$$X_{n+1} = (aX_n + c) \bmod m$$

then it is easy to see that the scheme will generate all even integers, all odd integers, or will alternate between even and odd integers, depending on the choice for a and c. Often, a and c are chosen to create a sequence of alternating even and odd integers. This has a tremendous impact on the simulation used for calculating π. The simulation depends on counting the number of pairs of integers whose greatest common divisor is 1. With truly random integers, one-fourth of the pairs should consist of two even integers, which of course have a gcd greater than 1. This never occurs with sequences that alternate between even and odd integers. To get the correct value of π using Cesaro's method, the number of pairs with a gcd of 1 should be approximately 60.8%. When pairs are used where one number is odd and the other even, this percentage comes out too high, around 80%, thus leading to the too small value of π. For a further discussion, see Danilowicz, R. "Demonstrating the Dangers of Pseudo-Random Numbers," *SIGCSE Bulletin*, June 1989.

7.10 a.

Pair	**Probability**
00	$(0.5 - \partial)^2 = 0.25 - \partial + \partial^2$
01	$(0.5 - \partial) \times (0.5 + \partial) = 0.25 - \partial^2$
10	$(0.5 + \partial) \times (0.5 - \partial) = 0.25 - \partial^2$
11	$(0.5 + \partial)^2 = 0.25 + \partial + \partial^2$

b. Because 01 and 10 have equal probability in the initial sequence, in the modified sequence, the probability of a 0 is 0.5 and the probability of a 1 is 0.5.

c. The probability of any particular pair being discarded is equal to the probability that the pair is either 00 or 11, which is $0.5 + 2\partial^2$, so the expected number of input bits to produce x output bits is $x/(0.25 - \partial^2)$.

d. The algorithm produces a totally predictable sequence of exactly alternating 1's and 0's.

7.11 a. For the sequence of input bits $a_1, a_2, \ldots, a_n$, the output bit b is defined as:

$$b = a_1 \oplus a_2 \oplus \ \ldots \oplus \ a_n$$

b. $0.5 - 2\partial^2$

c. $0.5 - 8\partial^4$

d. The limit as n goes to infinity is 0.5.

7.12 Yes. The eavesdropper is left with two strings, one sent in each direction, and their XOR is the secret key.

CHAPTER 8
INTRODUCTION TO NUMBER THEORY

ANSWERS TO QUESTIONS

8.1 An integer $p > 1$ is a prime number if and only if its only divisors are ± 1 and $\pm p$.

8.2 We say that a nonzero b divides a if $a = mb$ for some m, where a, b, and m are integers.

8.3 Euler's totient function, written $\phi(n)$, is the number of positive integers less than n and relatively prime to n.

8.4 The algorithm takes a candidate integer n as input and returns the result "composite" if n is definitely not a prime, and the result "inconclusive" if n may or may not be a prime. If the algorithm is repeatedly applied to a number and repeatedly returns inconclusive, then the probability that the number is actually prime increases with each inconclusive test. The probability required to accept a number as prime can be set as close to 1.0 as desired by increasing the number of tests made.

8.5 If r and n are relatively prime integers with $n > 0$. and if $\phi(n)$ is the least positive exponent m such that $a^m \equiv 1 \bmod n$, then r is called a primitive root modulo n.

8.6 The two terms are synonymous.

ANSWERS TO PROBLEMS

8.1 **a.** We are assuming that p_n is the largest of all primes. Because $X > p_n$, X is not prime. Therefore, we can find a prime number p_m that divides X.

b. The prime number p_m cannot be any of $p_1, p_2, \ldots, p_n$; otherwise p_m would divide the difference $X - p_1p_2\ldots p_n = 1$, which is impossible. Thus, $m > n$.

c. This construction provides a prime number outside any finite set of prime numbers, so the complete set of prime numbers is not finite.

d. We have shown that there is a prime number $>p_n$ that divides $X = 1 + p_1p_2\ldots p_n$, so p_{n+1} is equal to or less than this prime. Therefore, since this prime divides X, it is ≤ X and therefore $p_{n+1} \leq$ X.

8.2 **a.** gcd(a, b) = d if and only if a is a multiple of d and b is a multiple of d and gcd(a/d, b/d) = 1. The probability that an integer chosen at random is a multiple of d is just 1/d. Thus the probability that gcd(a, b) = d is equal to 1/d times 1/d times P, namely, P/d^2.

b. We have

$$\sum_{d\geq 1} \Pr[\gcd(a,b)=d] = \sum_{d\geq 1} \frac{P}{d^2} = P\sum_{d\geq 1}\frac{1}{d^2} = P\times\frac{\pi^2}{6} = 1$$

To satisfy this equation, we must have $P = \dfrac{6}{\pi^2}$ = 0.6079.

8.3 If p were any prime dividing n and n + 1 it would also have to divide

$$(n+1) - n = 1$$

8.4 Fermat's Theorem states that if p is prime and a is a positive integer not divisible by p, then $a^{p-1} \equiv 1 \pmod p$. Therefore $3^{10} \equiv 1 \pmod{11}$. Therefore $3^{201} = (3^{10})^{20} \times 3 \equiv 3 \pmod{11}$.

8.5 12

8.6 6

8.7 1

8.8 6

8.9 If a is one of the integers counted in $\phi(n)$, that is, one of the integers not larger than n and prime to n, the n – 1 is another such integer, because gcd(a, n) = gcd(m – a, m). The two integers, a and n – a, are distinct, because a = n – a gives n = 2a, which is inconsistent with the assumption that gcd(a, n) = 1. Therefore, for n > 2, the integers counted in $\phi(n)$ can be paired off, and so the number of them must be even.

8.10 Only multiples of p have a factor in common with p^n, when p is prime. There are just p^{n-1} of these $\leq p^n$, so $\phi(p^n) = p^n - p^{n-1}$.

8.11 **a.** $\phi(41) = 40$, because 41 is prime

b. $\phi(27) = \phi(3^3) = 3^3 - 3^2 = 27 - 9 = 18$

c. $\phi(231) = \phi(3) \times \phi(7) \times \phi(11) = 2 \times 6 \times 10 = 120$

d. $\phi(440) = \phi(2^3) \times \phi(5) \times \phi(11) = (2^3 - 2^2) \times 4 \times 10 = 160$

8.12 It follows immediately from the result stated in Problem 8.10.

8.13 totient

8.14 **a.** For n = 5, $2^n - 2 = 30$, which is divisible by 5.

b. We can rewrite the Chinese test as $(2^n - 2) \equiv 0 \bmod n$, or equivalently, $2^n \equiv 2 \pmod n$. By Fermat's Theorem, this relationship is true **if** n is prime (Equation 8.2).

c. For n = 15, $2^n - 2 = 32{,}766$, which is divisible by 15.

d. $2^{10} = 1024 \equiv 1 \pmod{341}$
$2^{340} = (2^{10})^{34} \equiv (1 \bmod 341)$
$2^{341} \equiv 2 \pmod{341}$

8.15 First consider a = 1. In step 3 of TEST(n), the test is **if** 1^q mod n = 1 **then** return("inconclusive"). This clearly returns "inconclusive." Now consider a = n – 1. In step 5 of TEST(n), for j = 0, the test is if $(n - 1)^q$ mod n = n – 1 **then** return("inconclusive"). This condition is met by inspection.

8.16 In Step 1 of TEST(2047), we set k = 1 and q = 1023, because $(2047 - 1) = (2^1)(1023)$. In Step 2 we select a = 2 as the base.
In Step 3, we have a^q mod n = 2^{1023} mod 2047 = $(2^{11})^{93}$ mod 2047 = $(2048)^{93}$ mod 2047 = 1 and so the test is passed.

8.17 There are many forms to this proof, and virtually every book on number theory has a proof. Here we present one of the more concise proofs. Define $M_i = M/m_i$. Because all of the factors of M are pairwise relatively prime, we have $\gcd(M_i, m_i) = 1$. Thus, there are solutions N_i of

$$N_i M_i \equiv 1 \pmod{m_i}$$

With these N_i, the solution x to the set of congruences is:

$$x \equiv a_1 N_1 M_1 + \ldots + a_k N_k M_k \pmod M$$

To see this, we introduce the notation $\langle x \rangle_m$, by which we mean the least positive residue of x modulo m. With this notation, we have

$$\langle x \rangle_{mi} \equiv a_i N_i M_i \equiv a_i \pmod{m_i}$$

because all other terms in the summation above that make up x contain the factor m_i and therefore do not contribute to the residue modulo m_i. Because $N_i M_i \equiv 1 \pmod{m_i}$, the solution is also unique modulo M, which proves this form of the Chinese Remainder Theorem.

8.18 We have $M = 3 \times 5 \times 7 = 105$; $M/3 = 35$; $M/5 = 21$; $M/7 = 15$.
The set of linear congruences

$$35b_1 \equiv 1 \pmod 3; \qquad 21b_2 \equiv 1 \pmod 5; \qquad 15b_3 \equiv 1 \pmod 7$$

has the solutions $b_1 = 2$; $b_2 = 1$; $b_3 = 1$. Then,

$$x \equiv 2 \times 2 \times 35 + 3 \times 1 \times 21 + 2 \times 1 \times 15 \equiv 233 \pmod{105} = 23$$

8.19 If the day in question is the xth (counting from and including the first Monday), then

$$x = 1 + 2K_1 = 2 + 3K_2 = 3 + 4K_3 = 4 + K_4 = 5 + 6K_5 = 6 + 5K_6 = 7K_7$$

where the K_i are integers; i.e.,

(1) $x \equiv 1 \bmod 2$; (2) $x \equiv 2 \bmod 3$; (3) $x \equiv 3 \bmod 4$; (4) $x \equiv 4 \bmod 1$; (5) $x \equiv 5 \bmod 6$; (6) $x \equiv 6 \bmod 5$; (7) $x \equiv 0 \bmod 7$

Of these congruences, (4) is no restriction, and (1) and (2) are included in (3) and (5). Of the two latter, (3) shows that x is congruent to 3, 7, or 11 (mod 12), and (5) shows the x is congruent to 5 or 11, so that (3) and (5) together are equivalent to $x \equiv 11 \pmod{12}$. Hence, the problem is that of solving:

$$x \equiv 11 \pmod{12}; \qquad x \equiv 6 \bmod 5; \quad x \equiv 0 \bmod 7$$

or

$$x \equiv -1 \pmod{12}; \qquad x \equiv 1 \bmod 5; \quad x \equiv 0 \bmod 7$$

Then $m_1 = 12$; $m_2 = 5$; $m_3 = 7$; $M = 420$
$M_1 = 35$; $M_2 = 84$; $M_3 = 60$
Then,

$$x \equiv (-1)(-1)35 + (-1)1 \times 21 + 2 \times 0 \times 60 = -49 \equiv 371 \pmod{420}$$

The first x satisfying the condition is 371.

8.20 2, 3, 8, 12, 13, 17, 22, 23

8.21 **a.** $x = 2, 27 \pmod{29}$
b. $x = 9, 24 \pmod{29}$
c. $x = 8, 10, 12, 15, 18, 26, 27 \pmod{29}$

CHAPTER 9
PUBLIC-KEY CRYPTOGRAPHY AND RSA

ANSWERS TO QUESTIONS

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

9.2 A user's private key is kept private and known only to the user. The user's public key is made available to others to use. The private key can be used to encrypt a signature that can be verified by anyone with the public key. Or the public key can be used to encrypt information that can only be decrypted by the possessor of the private key.

9.3 **Encryption/decryption:** The sender encrypts a message with the recipient's public key. **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message. **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

9.4 **1.** It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).

2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D(PR_b, E(PU_b, M))$$

4. It is computationally infeasible for an opponent, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an opponent, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

9.5 A **one-way function** is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy whereas the calculation of the inverse is infeasible:

9.6 A **trap-door one-way function** is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. With the additional information the inverse can be calculated in polynomial time.

9.7 **1.** Pick an odd integer n at random (e.g., using a pseudorandom number generator).
2. Pick an integer $a < n$ at random.
3. Perform the probabilistic primality test, such as Miller-Rabin. If n fails the test, reject the value n and go to step 1.
4. If n has passed a sufficient number of tests, accept n; otherwise, go to step 2.

Answers to Problems

9.1 This proof is discussed in the CESG report mentioned in Chapter 9 [ELLI99].

a. M3 =

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

b. Assume a plaintext message p is to be encrypted by Alice and sent to Bob. Bob makes use of M1 and M3, and Alice makes use of M2. Bob chooses a random number, k, as his private key, and maps k by M1 to get x, which he sends as his public key to Alice. Alice uses x to encrypt p with M2 to get z, the ciphertext, which she sends to Bob. Bob uses k to decrypt z by means of M3, yielding the plaintext message p.

c. If the numbers are large enough, and M1 and M2 are sufficiently random to make it impractical to work backwards, p cannot be found without knowing k.

9.2 **a.** $n = 33$; $\phi(n) = 20$; $d = 3$; C = 26.
b. $n = 55$; $\phi(n) = 40$; $d = 27$; C = 14.
c. $n = 77$; $\phi(n) = 60$; $d = 53$; C = 57.
d. $n = 143$; $\phi(n) = 120$; $d = 11$; C = 106.

e. $n = 527$; $\phi(n) = 480$; $d = 343$; C = 128. For decryption, we have

$$\begin{aligned} 128^{343} \bmod 527 &= 128^{256} \times 128^{64} \times 128^{16} \times 128^{4} \times 128^{2} \times 128^{1} \bmod 527 \\ &= 35 \times 256 \times 35 \times 101 \times 47 \times 128 = 2 \bmod 527 \\ &= 2 \bmod 257 \end{aligned}$$

9.3 5

9.4 By trail and error, we determine that $p = 59$ and $q = 61$. Hence $\phi(n) = 58 \text{ x } 60 = 3480$. Then, using the extended Euclidean algorithm, we find that the multiplicative inverse of 31 modulu $\phi(n)$ is 3031.

9.5 Suppose the public key is n = pq, e. Probably the order of e relative to (p – 1)(q – 1) is small so that a small power of e gives us something congruent to 1 mod(p – 1)(q – 1). In the worst case where the order is 2 then e and d (the private key) are the same. Example: if p = 7 and q = 5 then (p – 1)(q – 1) = 24. If e = 5 then e squared is congruent to 1 mod(p – 1)(q – 1); that is, 25 is congruent to 24 mod 1.

9.6 Yes. If a plaintext block has a common factor with n modulo n then the encoded block will also have a common factor with n modulo n. Because we encode blocks, which are smaller than pq, the factor must be p or q and the plaintext block must be a multiple of p or q. We can test each block for primality. If prime, it is p or q. In this case we divide into n to find the other factor. If not prime, we factor it and try the factors as divisors of n.

9.7 No, it is not safe. Once Bob leaks his private key, Alice can use this to factor his modulus, N. Then Alice can crack any message that Bob sends.

Here is one way to factor the modulus:

Let k= ed – 1. Then k is congruent to 0 mod ϕ(N) (where 'ϕ' is the Euler totient function). Select a random x in the multiplicative group Z(N). Then $x^k \equiv 1 \bmod N$, which implies that $x^{k/2}$ is a square root of 1 mod N. With 50% probability, this is a nontrivial square root of N, so that

$\gcd(x^{k/2} - 1, N)$ will yield a prime factor of N.

If $x^{k/2} = 1 \bmod N$, then try $x^{k/4}$, $x^{k/8}$, etc...

This will fail if and only if $x^{k/2^i} \equiv -1$ for some i. If it fails, then choose a new x.

This will factor N in expected polynomial time.

9.8 Consider a set of alphabetic characters {A, B, …, Z}. The corresponding integers, representing the position of each alphabetic character in the alphabet, form a set of message block values SM = {0, 1, 2, …, 25}. The set of corresponding ciphertext block values SC = {0^e mod N, 1^e mod N, …, 25^e mod N}, and can be computed by everybody with the knowledge of the public key of Bob.

Thus, the most efficient attack against the scheme described in the problem is to compute M^e mod N for all possible values of M, then create a look-up table with a ciphertext as an index, and the corresponding plaintext as a value of the appropriate location in the table.

9.9 **a.** We consider n = 233, 235, 237, 239, and 241, and the base a = 2:

n = 233

$233 - 1 = 2^3 \times 29$, thus k=3, q=29

a^q mod n = 2^{29} mod 233 = 1

test returns "inconclusive" ("probably prime")

n = 235

$235 - 1 = 2^1 \times 117$, thus k=1, q=117

a^q mod n = 2^{117} mod 235 = 222

$222 \neq 1$ and $222 \neq 235 - 1$

test returns "composite"

n = 237

$237 - 1 = 2^2 \times 59$, thus k=2, q=59

a^q mod n = 2^{59} mod 237 = $167 \neq 1$

$167 \neq 237 - 1$

167^2 mod 237 = $160 \neq 237 - 1$

test returns "composite"

n = 239

$239 - 1 = 2^1 \times 119$.

2^{119} mod 239 = 1

test returns "inconclusive" ("probably prime")

n = 241

$241 - 1 = 2^4 \times 15$

2^4 mod 241 = 16

$16 \neq 1$ and $16 \neq 241 - 1$

16^2 mod 241 = 256 mod 241 = 15

$15 \neq 241 - 1$

15^2 mod 241 = 225 mod 241 = 225

$225 \neq 241 - 1$

225^2 mod 241 = 15

$15 \neq 241 - 1$

test returns "inconclusive" ("probably prime")

b. M=2, e=23, n=233 × 241=56,153 therefore p=233 and q=241

e = 23 = (10111)2

I		4	3	2	1	0

e_i		1	0	1	1	1
D	1	2	4	32	2048	**21,811**

c. Compute private key (d, p, q) given public key (e=23, n=233 × 241=56,153).
Since n=233 × 241=56,153, p=233 and q=241
$\phi(n) = (p - 1)(q - 1) = 55,680$
Using Extended Euclidean algorithm, we obtain
$d = 23^{-1} \bmod 55680 = 19,367$

d. Without CRT: $M = 21,811^{19,367} \bmod 56,153 = 2$
With CRT:
$d_p = d \bmod (p - 1)$ $\qquad$ $d_q = d \bmod (q-1)$
$d_p = 19367 \bmod 232 = 111$ $\qquad$ $d_q = 19367 \bmod 240 = 167$

$C_p = C \bmod p$
$M_p = C_p^{d_p} \bmod p = 141^{111} \bmod 233 =2$
$C_q = C \bmod q$
$M_q = C_q^{d_q} \bmod q$
$M_q = 121^{167} \bmod 241 = 2$
M = 2.

9.10 $C = (M^{d_S} \bmod NS)^{e_R} \bmod NR = S^{e_R} \bmod NR$
where
$S = M^{d_S} \bmod NS.$
$M' = (C^{d_R} \bmod NR)^{e_S} \bmod NS = S'^{e_S} \bmod NS =$
where
$S' = C^{d_R} \bmod NR.$

The scheme does not work correctly if $S \neq S'$. This situation may happen for a significant subset of messages M if $N_S > N_R$. In this case, it might happen that $N_R \leq S < N_S$, and since by definition $S' < N_R$, then $S \neq S'$, and therefore also $M' \neq M$. For all other relations between N_S and N_R, the scheme works correctly (although $N_S = N_R$ is discouraged for security reasons).

In order to resolve the problem both sides can use two pairs of keys, one for encryption and the other for signing, with all signing keys N_{SGN} smaller than the encryption keys N_{ENC}

9.11 3rd element, because it equals to the 1st squared,
5th element, because it equals to the product of 1st and 2nd
7th element, because it equals to the cube of 1st,
etc.

9.12 Refer to Figure 9.5 The private key k is the pair {d, n}; the public key x is the pair {e, n}; the plaintext p is M; and the ciphertext z is C. M1 is formed by calculating d = e^{-1} mod ϕ(n). M2 consists of raising M to the power e (mod n). M2 consists of raising C to the power d (mod n).

9.13 Yes.

9.14 This algorithm is discussed in the CESG report mentioned in Chapter 6 [ELLI99], and is known as Cocks algorithm.

a. Cocks makes use of the Chinese remainder theorem (see Section 8.4 and Problem 8.10), which says it is possible to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli. In particular for relatively prime P and Q, any integer M in the range $0 \leq M < N$ can be the pair of numbers M mod P and M mod Q, and that it is possible to recover M given M mod P and M mod Q. The security lies in the difficulty of finding the prime factors of N.

b. In RSA, a user forms a pair of integers, d and e, such that de ≡ 1 mod ((P – 1)(Q – 1)), and then publishes e and N as the public key. Cocks is a special case in which e = N.

c. The RSA algorithm has the merit that it is symmetrical; the same process is used both for encryption and decryption, which simplifies the software needed. Also, e can be chosen arbitrarily so that a particularly simple version can be used for encryption with the public key. In this way, the complex process would be needed only for the recipient.

d. The private key k is the pair P and Q; the public key x is N; the plaintext p is M; and the ciphertext z is C. M1 is formed by multiplying the two parts of k, P and Q, together. M2 consists of raising M to the power N (mod N). M3 is the process described in the problem statement.

9.15 1) Adversary X intercepts message sent by A to B, i.e. [A, E(PU_b, M), B]

2) X sends B [X, E(PU_b, M), B]

3) B acknowledges receipt by sending X [B, E(PU_x, M), X]

4) X decrypts E(PU_x, M) using his secret decryption key, thus getting M

9.16

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	1	0	1	0	1	0	0
c	1	2	4	5	11	23	46	93	186	372
f	5	25	625	937	595	569	453	591	59	1013

9.17 First, let us consider the algorithm in Figure 9.7. The binary representation of b is read from left to right (most significant to least significant) to control which operations are performed. In essence, if c is the current value of the exponent after some of the bits have been processed, then if the next bit is 0, the exponent is doubled (simply a left shift of 1 bit) or it is doubled and incremented by 1. Each iteration of the loop uses one of the identities:

$$a^{2c} \bmod n = \left(a^c\right)^2 \bmod n \qquad \text{if } b_i = 0$$
$$a^{2c+1} \bmod n = a \times \left(a^c\right)^2 \bmod n \quad \text{if } b_i = 1$$

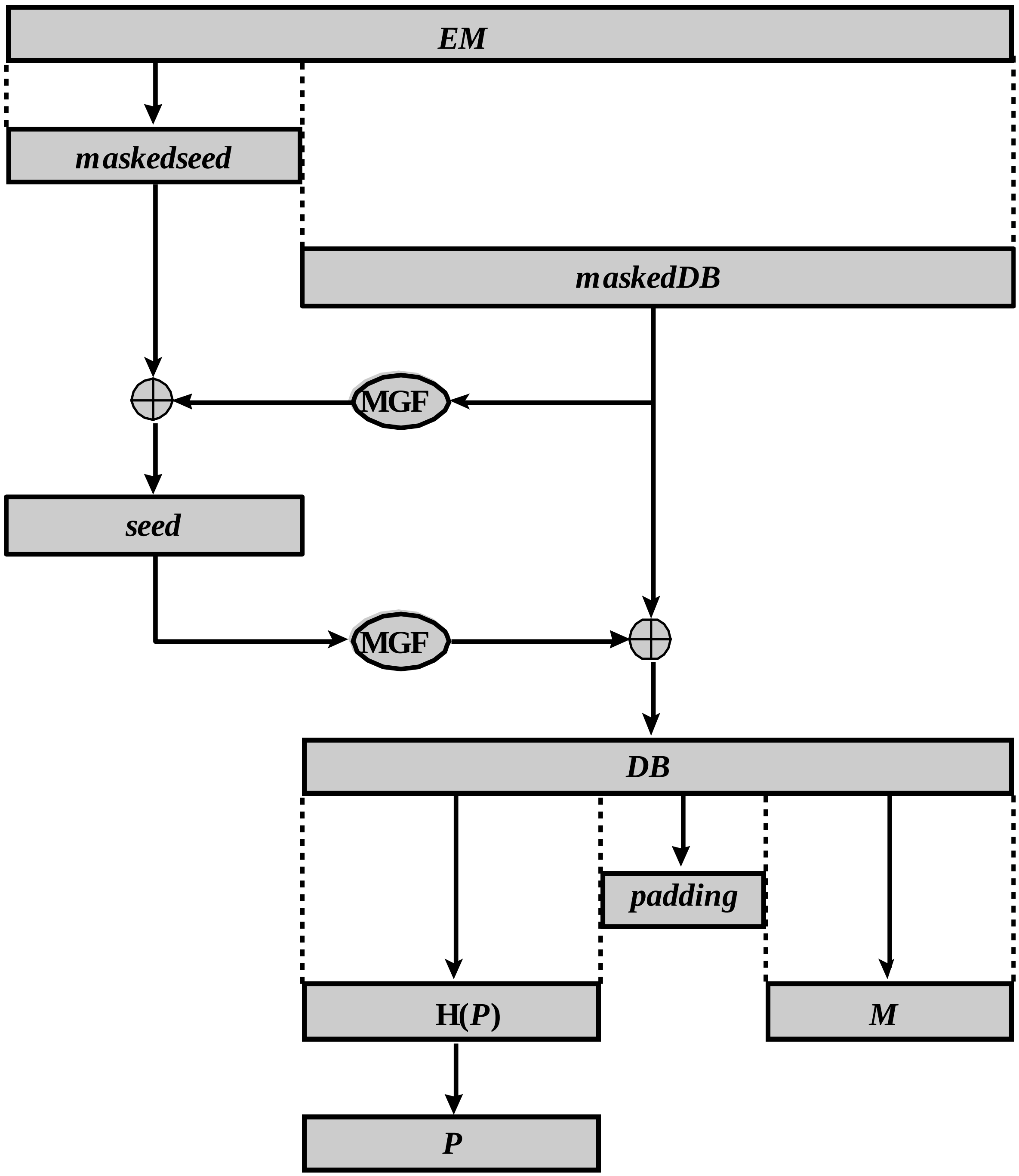
The algorithm preserves the invariant that $d = a^c \bmod n$ as it increases c by doublings and incrementations until $c = b$.

Now let us consider the algorithm in the problem, which is adapted from one in [KNUT98, page 462]. This algorithm processes the binary representation of b from right to left (least significant to most significant). In this case, the algorithm preserves the invariant that $a^n = d \times T^E$. At the end, E = 0, leaving $a^n = d$.

9.18 Note that because $Z = r^e \bmod n$, then $r = Z^d \bmod n$. Bob computes:

$$tY \bmod n = r^{-1}X^d \bmod n = r^{-1}Z^dC^d \bmod n = C^d \bmod n = M$$

9.19



9.20 a. By noticing that $x^{i+1} = x^i \times x$, we can avoid a large amount of recomputation for the S terms.

> **algorithm** P2;
> n, i: integer; x, polyval: real;
> a, S, power: **array** [0..100] **of** real;
> **begin**
> read(x, n);
> power[0] := 1; read(a[0]); S[0] := a[0];
> **for** i := 1 **upto** n **do**

```
        begin
            read(a[i]); power[i] := x × power[i – 1];
            S[i] := a[i] × power[i]
        end;
        polyval := 0;
        for i ;= 0 upto n do polyval := polyval + S[i];
        write ('value at', x, 'is', polyval)
end.
```

b. The hint, known as Horner's rule, can be written in expanded form for P(x):

$$P(x) = ((\ldots(a_n x + a_{n-1})x + a_{n-2})x + \ldots + a_1) + a_0$$

We use this to produce the revised algorithm:

```
algorithm P2;
    n, i: integer; x, polyval: real;
    a: array [0..100] of real;
    begin
        read(x, n);
        polyval := 0;
        for i := 0 upto n do
        begin
            read(a[n – i]); polyval := polyval × x × a[n – 1]
        end;
        write ('value at', x, 'is', polyval)
end.
```

P3 is a substantial improvement over P2 not only in terms of time but also in terms of storage requirements.

9.21 $90 + 455 + 341 + 132 + 56 + 82 = 1.156 \times 10^3$

9.22 **a.** $w^{-1} \equiv 3 \pmod{20}$; **a** = (7, 1, 15, 10); ciphertext = 18.
b. $w^{-1} \equiv 387 \pmod{491}$; **a** = (203, 118, 33, 269, 250, 9, 112, 361); ciphertext = 357.
c. $w^{-1} \equiv 15 \pmod{53}$; **a** = (39, 32, 11, 22, 37); ciphertext = 119.
d. $w^{-1} \equiv 1025 \pmod{9291}$; **a** = (8022, 6463, 7587, 7986, 65, 8005, 6592, 7274); ciphertext = 30869.

9.23 To see this requirement, let us redo the derivation Appendix F, expanding the vectors to show the actual arithmetic.
The sender develops a simple knapsack vector **a**' and a corresponding hard knapsack **a** = w**a**' mod m. To send a message x, the sender computes and sends:

$$S = \mathbf{a} \bullet \mathbf{x} = \sum a_i x_i$$

Now, the receiver can easily compute S' and solve for **x**:

$$\begin{aligned}
S' &= w^{-1} S \bmod m \\
&= w^{-1} \sum a_i x_i \bmod m \\
&= w^{-1} \sum \left(w a'_i \bmod m \right) x_i \bmod m \\
&= \sum \left(w^{-1} w a'_i \bmod m \right) x_i \\
&= \sum a'_i x_i \bmod m
\end{aligned}$$

Each of the xi has a value of zero or one, so that the maximum value of the summation is $\sum a_i$. If $m > \sum a_i$, then the mod m term has no effect and we have

$$S' = \sum a'_i x_i$$

This can easily be solved for the x_i.

Chapter 10
Key Management; Other Public-Key Cryptosystems

Answers to Questions

10.1 **1.** The distribution of public keys. **2.** The use of public-key encryption to distribute secret keys

10.2 Public announcement. Publicly available directory. Public-key authority. Public-key certificates

10.3 **1.** The authority maintains a directory with a {name, public key} entry for each participant. **2.** Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication. **3.** A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way. **4.** Periodically, the authority publishes the entire directory or updates to the directory. For example, a hard-copy version much like a telephone book could be published, or updates could be listed in a widely circulated newspaper. **5.** Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

10.4 A public-key certificate contains a public key and other information, is created by a certificate authority, and is given to the participant with the matching private key. A participant conveys its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority.

10.5 **1.** Any participant can read a certificate to determine the name and public key of the certificate's owner. **2.** Any participant can verify that the certificate originated from the certificate authority and is not counterfeit. **3.** Only the certificate authority can create and update certificates. **4.** Any participant can verify the currency of the certificate.

10.6 Two parties each create a public-key, private-key pair and communicate the public key to the other party. The keys are designed in such a way that both sides can calculate the same unique secret key based on each side's private key and the other side's public key.

10.7 An elliptic curve is one that is described by cubic equations, similar to those used for calculating the circumference of an ellipse. In general, cubic equations for elliptic curves take the form

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

where a, b, c, d, and e are real numbers and x and y take on values in the real numbers

10.8 Also called the point at infinity and designated by O. This value serves as the additive identity in elliptic-curve arithmetic.

10.9 If three points on an elliptic curve lie on a straight line, their sum is O.

Answers to Problems

10.1 **a.** $Y_A = 7^5 \bmod 71 = 51$
b. $Y_B = 7^{12} \bmod 71 = 4$
c. $K = 4^5 \bmod 71 = 30$

10.2 **a.** $\phi(11) = 10$
$2^{10} = 1024 = 1 \bmod 11$
If you check 2^n for $n < 10$, you will find that none of the values is 1 mod 11.
b. 6, because $2^6 \bmod 11 = 9$
c. $K = 3^6 \bmod 11 = 3$

10.3 For example, the key could be $x_A^g x_B^g = (x_A x_B)^g$. Of course, Eve can find that trivially just by multiplying the public information. In fact, no such system could be secure anyway, because Eve can find the secret numbers x_A and x_B by using Fermat's Little Theorem to take g-th roots.

10.4 $x_B = 3$, $x_A = 5$, the secret combined key is $(3^3)^5 = 3^15 = 14348907$.

10.5 **1.** Darth prepares for the attack by generating a random private key X_D and then computing the corresponding public key Y_D.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_D to Bob. Darth also calculates
$K2 = (Y_A)^{X_D} \bmod q$

4. **Bob** receives Y_D and calculates $K1 = (Y_D)^{X_B} \bmod q$.
5. **Bob** transmits X_A to Alice.
6. **Darth** intercepts X_A and transmits Y_D to Alice. Darth calculates $K1 = (Y_B)^{X_D} \bmod q$.
7. **Alice** receives Y_D and calculates $K2 = (Y_D)^{X_A} \bmod q$.

10.6 From Figure 10.7, we have, for private key X_B, B's public key is $Y_B = \alpha^{X_B} \bmod q$.

1. User B computes $(C_1)^{X_B} \bmod q = \alpha^{kX_B} \bmod q$.
   But $K = (Y_B)^k \bmod q = \left(\alpha^{X_B} \bmod q\right)^k \bmod q = \alpha^{kX_B} \bmod q$
   So step 1 enables user B to recover K.
2. Next, user B computes $\left(C_2K^{-1}\right) \bmod q = \left(KMK^{-1}\right) \bmod q = M$, which is the desired plaintext.

10.7 **a.** (49, 57)
b. $C_2 = 29$

10.8 **a.** For a vertical tangent line, the point of intersection is infinity. Therefore $2Q = O$.
b. $3Q = 2Q + Q = O + Q = Q$.

10.9 We use Equation (10.1), which defines the form of the elliptic curve as $y^2 = x^3 + ax + b$, and Equation (10.2), which says that an elliptic curve over the real numbers defines a group if $4a^3 + 27b^2 \neq 0$.
a. For $y^2 = x^3 - x$, we have $4(-1)^3 + 27(0) = -4 \neq 0$.
b. For $y^2 = x^3 + x + 1$, we have $4(1)^3 + 27(1) = 21 \neq 0$.

10.10 Yes, since the equation holds true for $x = 4$ and $y = 7$:

$7^2 = 4^3 - 5(4) + 5$
$49 = 64 - 20 + 5 = 49$

10.11 **a.** First we calculate $R = P + Q$, using Equations (10.3).

$\Delta = (8.5 - 9.5)/(-2.5 + 3.5) = -1$
$x_R = 1 + 3.5 + 2.5 = 7$
$y_R = -8.5 - (-3.5 - 7) = 2$
$R = (7, 2)$

b. For $R = 2P$, we use Equations (10.4), with $a = -36$

$x_r = [(36.75 - 36)/19]^2 + 7 \approx 7$
$y_R = [(36.75 - 36)/19](-3.5 - 7) - 9.5 \approx 9.9$

10.12 $(4a^3 + 27b^2) \bmod p = 4(10)^3 + 27(5)^2 \bmod 17 = 4675 \bmod 17 = 0$

This elliptic curve does not satisfy the condition of Equation (10.6) and therefore does not define a group over Z_{17}.

10.13

x	$(x^3 + x + 6) \bmod 11$	square roots mod p?	y
0	6	no	
1	8	no	
2	5	yes	4, 7
3	3	yes	5, 6
4	8	no	
5	4	yes	2, 9
6	8	no	
7	4	yes	2, 9
8	9	yes	3, 8
9	7	no	
10	4	yes	2, 9

10.14 The negative of a point $P = (x_P, y_P)$ is the point $-P = (x_P, -y_P \bmod p)$. Thus $-P = (5,9)$; $-Q = (3,0)$; $-R = (0,11)$

10.15 We follow the rules of addition described in Section 10.4. To compute 2G = (2, 7) + (2, 7), we first compute

$$\begin{aligned}\lambda &= (3 \times 2^2 + 1)/(2 \times 7) \bmod 11 \\ &= 13/14 \bmod 11 = 2/3 \bmod 11 = 8\end{aligned}$$

Then we have

$$\begin{aligned}&x_3 = 8^2 - 2 - 2 \bmod 11 = 5 \\ &y_3 = 8(2 - 5) - 7 \bmod 11 = 2 \\ &2G = (5, 2)\end{aligned}$$

Similarly, 3G = 2G + G, and so on. The result:

2G = (5, 2)	3G = (8, 3)	4G = (10, 2)	5G = (3, 6)
6G = (7, 9)	7G = (7, 2)	8G = (3, 5)	9G = (10, 9)
10G = (8, 8)	11G = (5, 9)	12G = (2, 4)	13G = (2, 7)

10.16 **a.** $P_B = n_B \times G = 7 \times (2, 7) = (7, 2)$. This answer is seen in the preceding table.

b. $C_m = \{kG, P_m + kP_B\}$
$= \{3(2, 7), (10, 9) + 3(7, 2)\} = \{(8,3), (10, 9) + (3, 5)\} = \{(8, 3), (10, 2)\}$

c. $P_m = (10, 2) - 7(8, 3) = (10, 2) - (3, 5) = (10, 2) + (3, 6) = (10, 9)$

10.17 **a.** $S + kY_A = M - kx_AG + kx_AG = M$.

b. The imposter gets Alice's public verifying key Y_A and sends Bob M, k, and $S = M - kY_A$ for any k.

10.18 **a.** $S + kY_A = M - x_AC_1 + kY_A = M - x_AkG + kx_AG = M$.

b. Suppose an imposter has an algorithm that takes as input the public G, $Y_A = x_AG$, Bob's $C_1 = kG$, and the message M and returns a valid signature which Bob can verify as $S = M - kY_A$ and Alice can reproduce as $M - x_AC_1$. The imposter intercepts an encoded message $C_m = \{k'G', P_m + k'P_A\}$ from Bob to Alice where $P_A = n_AG'$ is Alice's public key. The imposter gives the algorithm the input $G = G'$, $Y_A = P_A$, $C_1 = k'G'$, $M = P_m + k'P_A$ and the algorithm computes an S which Alice could "verify" as $S = P_m + k'P_A - n_Ak'G' = P_m$.

c. Speed, likelihood of unintentional error, opportunity for denial of service or traffic analysis.

CHAPTER 11
MESSAGE AUTHENTICATION AND HASH FUNCTIONS

ANSWERS TO QUESTIONS

11.1 **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.

11.2 At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

11.3 Message encryption, message authentication code, hash function.

11.4 Error control code, then encryption.

11.5 An authenticator that is a cryptographic function of both the data to be authenticated and a secret key.

11.6 A hash function, by itself, does not provide message authentication. A secret key must be used in some fashion with the hash function to produce authentication. A MAC, by definition, uses a secret key to calculated a code used for authentication.

11.7 Figure 11.5 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows: **a.** The message plus concatenated hash code is encrypted using symmetric encryption. **b.** Only the hash code is encrypted, using symmetric encryption. **c.** Only the hash code is encrypted, using public-key encryption and using the sender's private key. **d.** If confidentiality as well as a digital signature is desired, then the message plus the public-key-encrypted hash code can be encrypted using a symmetric secret key. **e.** This technique uses a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify. **f.** Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

11.8 No. Section 11.3 outlines such attacks.

11.9
1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the **one-way** property.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

11.10 Property 5 in Question 11.9 defines **weak collision resistance**. Property 6 defines **strong collision resistance**.

11.11 A typical hash function uses a compression function as a basic building block, and involves repeated application of the compression function.

ANSWERS TO PROBLEMS

11.1 No. If internal error control is used, error propagation in the deciphering operation introduces too many errors for the error control code to correct.

11.2 The CBC mode with an IV of 0 and plaintext blocks D1, D2, . . ., Dn and 64-bit CFB mode with IV = D1 and plaintext blocks D2, D3, . . ., Dn yield the same result.

11.3 a. Yes. The XOR function is simply a vertical parity check. If there is an odd number of errors, then there must be at least one column that contains an odd number of errors, and the parity bit for that column will detect the error. Note that the RXOR function also catches all errors caused by an odd number of error bits. Each RXOR bit is a function of a unique "spiral" of bits in the block of data. If there is an odd number of errors, then there must be at least one spiral that contains an odd number of errors, and the parity bit for that spiral will detect the error.

b. No. The checksum will fail to detect an even number of errors when both the XOR and RXOR functions fail. In order for both to fail, the pattern of error bits must be at intersection points between parity spirals and parity columns such that there is an even number of error bits in each parity column and an even number of error bits in each spiral.

c. It is too simple to be used as a secure hash function; finding multiple messages with the same hash function would be too easy.

11.4 a. For clarity, we use overbars for complementation. We have:

$$\mathrm{E}\left(\overline{M_i},\overline{H_{i-1}}\right)=\overline{\mathrm{E}\left(M_i,H_{i-1}\right)}\oplus\overline{H_{i-1}}=\mathrm{E}\left(M_i,H_{i-1}\right)\oplus H_{i-1}$$

Therefore, the hash function of message M with initial value I is the same as the hash function for message N with initial value $\overline{I}$ for any given I, where

$$M=M_1\,||\,M_2\,||\ldots||\,M_n;\quad N=\overline{M_1}\,||\,M_2\,||\ldots||\,M_n$$

b. The same line of reasoning applies with the *M*s and *H*s reversed in the derivation.

11.5 a. It satisfies properties 1 through 3 but not the remaining properties. For example, for property 4, a message consisting of the value h satisfies $\mathrm{H}(h) = h$. For property 5, take any message M and add the decimal digit 0 to the sequence; it will have the same hash value.

b. It satisfies properties 1 through 3. Property 4 is also satisfied if n is a large composite number, because taking square roots modulo such an integer n is considered to be infeasible. Properties 5 and 6 are not satisfied because –M will have the same value as M.

c. 955

11.6 If you examine the structure of a single round of DES, you see that the round includes a one-way function, f, and an XOR:

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

For DES, the function f is depicted in Figure 3.5. It maps a 32-bit R and a 48-bit K into a 32-bit output. That is, it maps an 80-bit input into a 32-bit output. This is clearly a one-way function. Any hash function that produces a 32-bit output could be used for f. The demonstration in the text that decryption works is still valid for any one-way function f.

11.7 The opponent has the two-block message B1, B2 and its hash RSAH(B1, B2). The following attack will work. Choose an arbitrary C1 and choose C2 such that:

$$C2 = RSA(C1) \oplus RSA(B1) \oplus B2$$

then

$$\begin{aligned} RSA(C1) \oplus C2 &= RSA(C1) \oplus RSA(C1) \oplus RSA(B1) \oplus B2 \\ &= RSA(B1) \oplus B2 \end{aligned}$$

so

$$\begin{aligned} RSAH(C1, C2) &= RSA[RSA(C1) \oplus C2)] = RSA[RSA(B1) \oplus B2] \\ &= RSAH(B1, B2) \end{aligned}$$

11.8 The statement is false. Such a function cannot be one-to-one because the number of inputs to the function is of arbitrary, but the number of unique outputs is 2^n. Thus, there are multiple inputs that map into the same output.

CHAPTER 12
HASH AND MAC ALGORITHMS

ANSWERS TO QUESTIONS

12.1 In **little-endian format**, the least significant byte of a word is in the low-address byte position. In **big-endian format**, the most significant byte of a word is in the low-address byte position.

12.2 Addition modulo 2^{64} or 2^{32}, circular shift, primitive Boolean functions based on AND, OR, NOT, and XOR.

12.3 XOR, addition over a finite field, and circular shifts.

12.4 **1.** Cryptographic hash functions such as MD5 and SHA generally execute faster in software than symmetric block ciphers such as DES. **2.** Library code for cryptographic hash functions is widely available.

12.5 To replace a given hash function in an HMAC implementation, all that is required is to remove the existing hash function module and drop in the new module.

ANSWERS TO PROBLEMS

12.1 Assume an array of sixteen 64-bit words W[0], . . ., W[15], which will be treated as a circular queue. Define MASK = 0000000F in hex. Then for round t:

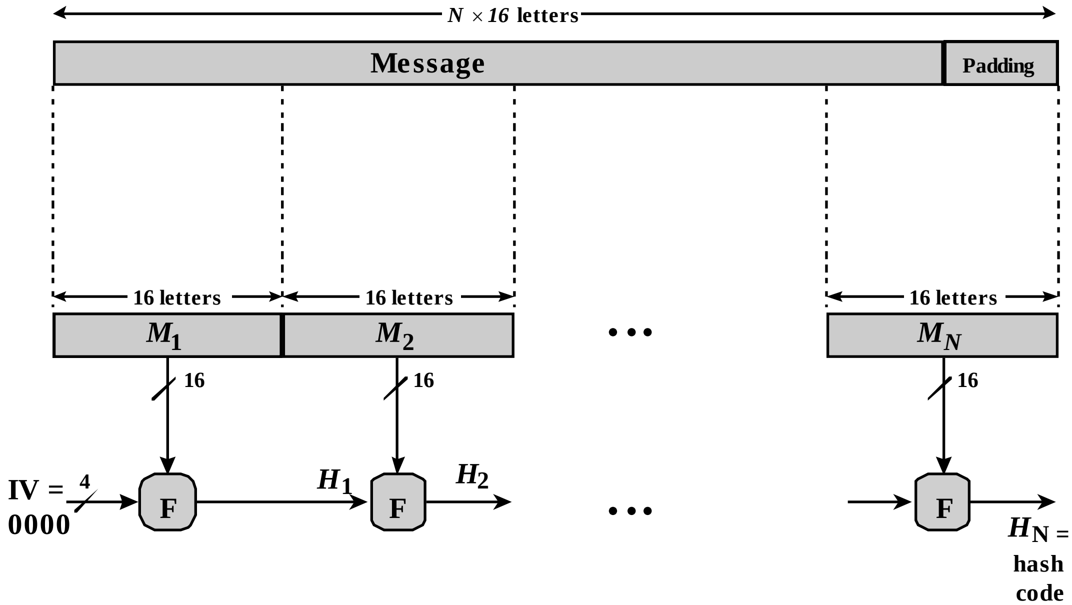
$s = t \wedge \text{MASK};$
if $(t \geq 16)$ then
$W[s] = W[s] \oplus \sigma_0(W[(s + 1) \wedge \text{MASK}]) \oplus$
$\quad W[(s + 9) \wedge \text{MASK}] \oplus \sigma_1(W[(s + 14] \wedge \text{MASK}])$

12.2 $W_{16} = W_0 \oplus \sigma_0(W_1) \oplus W_9 \oplus \sigma_1(W_{14})$
$W_{17} = W_1 \oplus \sigma_0(W_2) \oplus W_{10} \oplus \sigma_1(W_{15})$
$W_{18} = W_2 \oplus \sigma_0(W_3) \oplus W_{11} \oplus \sigma_1(W_{16})$
$W_{19} = W_3 \oplus \sigma_0(W_4) \oplus W_{12} \oplus \sigma_1(W_{17})$

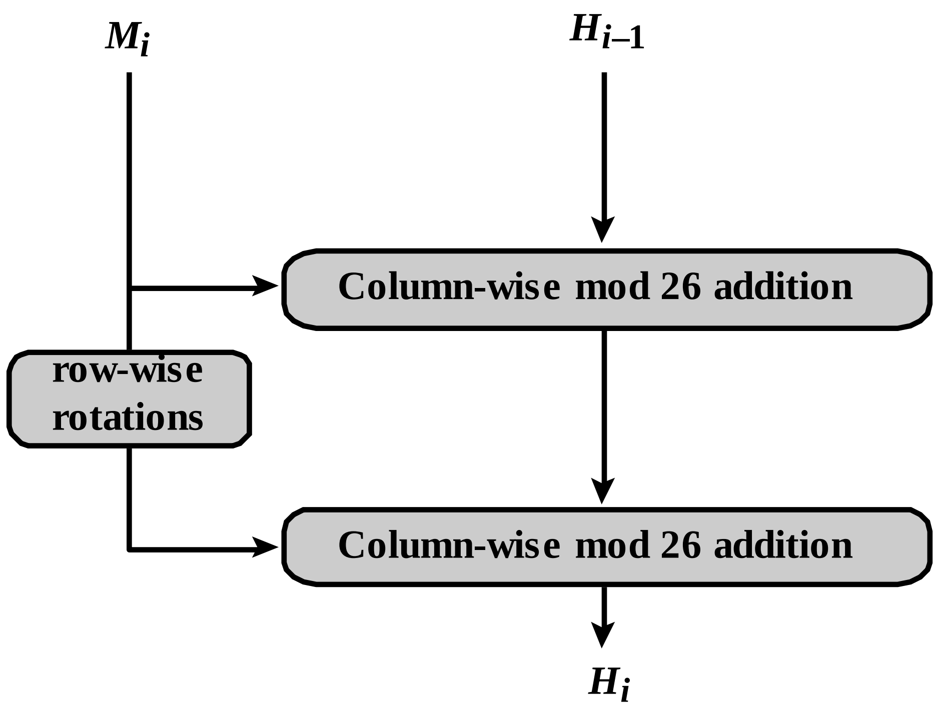
12.3 **a.** **1.** Interchange x_1 and x_4; x_2 and x_3; y_1 and y_4; and y_2 and y_3.
2. Compute $Z = X + Y \bmod 2^{32}$.
3. Interchange z_1 and z_4; and z_2 and z_3.

b. You must use the same sort of interchange.

12.4 **a.** Overall structure:



Compression function F:



b. BFQG

c. Simple algebra is all you need to generate a result:

AYHGDAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAA

12.5 Generator for GF(2^8) using $x^8 + x^4 + x^3 + x^2 + 1$. Partial results:

Power Representation	**Polynomial Representation**	**Binary Representation**	**Decimal (Hex) Representation**
0	0	00000000	00
g^0 (= g^{127})	1	00000001	01
g^1	g	00000010	02
g^2	g^2	00000100	04
g^3	g^3	00001000	08
g^4	g^4	00010000	10
g^5	g^5	00100000	20
g^6	g^6	01000000	40
g^7	g^7	10000000	80
g^8	$g^4 + g^3 + g^2 + 1$	00011101	1D
g^9	$g^5 + g^4 + g^3 + g$	00111010	3A
g^{10}	$g^6 + g^5 + g^4 + g^2$	01110100	74
g^{11}	$g^7 + g^6 + g^5 + g^3$	11101000	E8
g^{12}	$g^7 + g^6 + g^3 + g^2 + 1$	11001101	CD
g^{13}	$g^7 + g^2 + g + 1$	10000111	87
g^{14}	$g^4 + g + 1$	00010011	13

12.6

	00	01	10	11
00	1	B	9	C
01	D	6	F	3
10	E	8	7	4
11	A	2	5	0

E box

	00	01	10	11
00	F	0	D	7
01	B	E	5	A
10	9	2	C	1
11	3	4	8	6

E^{-1} box

12.7 **a.** For input 00: The output of the first E box is 0001. The output of the first E^{-1} box is 1111. The input to R is 1110 and the output of R is 0001. The input to the second E box is 0000 and the output is 0001. The input to the second E^{-1} box is 1110 and the output is 1000. So the final output is 00011000 in binary, which is 18 in hex. This agrees with Table 12.3a.

b. For input 55: The output of the first E box is 0110. The output of the first E^{-1} box is 1110. The input to R is 1000 and the output of R is 0110. The input to the second E box is 0000 and the output is 0001. The input to the second E^{-1} box is 1000 and the output is 1001. So the final output is 00011001 in binary, which is 19 in hex. This agrees with Table 12.3a.

c. For input 1E: The output of the first E box is 1011. The output of the first E^{-1} box is 1000. The input to R is 0011 and the output of R is 1101. The input to the second E box is 0110 and the output is 1111. The input to the second E^{-1} box is 0101 and the output is 1110. So the final output is 1111110 in binary, which is in hex FE. This agrees with Table 12.3a.

12.8 Treat the input to the S-box as two 4-bit variables u and v and the output as the 4-bit variables u' and v'. The S-box can be expressed as $(u', v') = S(u, v)$. Using Figure 12.9, we can express this as:

$$u' = E[E(u) \oplus r], \quad v' = E^{-1}[E^{-1}(v) \oplus r]$$

where $r = R[E(u) \oplus E^{-1}(v)]$

12.9 Consider the encryption $E(H_{i-1}, M_i)$. We could write the last round key as $K_{10} = E(RC, H_{i-1})$; this quantity is XORed onto the cipher state as the last encryption step. Now take a look at the recursion: $H_i = E(H_{i-1}, M_i) \oplus M_i$. Formally applying this construction to the "key encryption line" we get $K'_{10} = E(RC, H_{i-1}) \oplus H_{i-1}$. Using this value as the effective last round key formally creates two interacting lines (as compared to the interacting encryption lines), and results in the Whirlpool scheme, which therefore shows up as the natural choice for the compression function. This explanation is taken from the Whirlpool document.

12.10 We use the definition from Section 11.3. For a one-block message, the MAC using CBC-MAC is $T = E(K, X)$, where K is the key and X is the message block. Now consider the two-block message in which the first block is X and the second block is $X \oplus T$. Then the MAC is $E(K, [T \oplus \square\, X \oplus T\square]) = E(K, X) = T$.

12.11 We use Figure 12.12a but put the XOR with K_1 after the final encryption. For this problem, there are two blocks to process. The output of the encryption of the first message block is $E(K, \mathbf{0}) = CBC(K, \mathbf{0}) = T_0 \oplus K_1$. This is XORed with the second message block $(T_0 \oplus T_1)$, so that the input to the second encryption is $(T_1 \oplus K_1) = CBC(K, \mathbf{1}) = E(K, \mathbf{1})$. So the output of the second encryption is $E(K, [E(K, \mathbf{1})]) = CBC(K, [CBC(K, \mathbf{1})]) = T_2 \oplus K_1$. After the final XOR with K_1, we get $VMAC(K, [\mathbf{0} \,\|\, (T_0 \oplus T_1)]) = T_2$.

12.12 **a.** In each case (64 bits, 128 bits) the constant is the binary representation of the irreducible polynomial defined in Section 12.4. The two constants are

$$R_{128} = 0^{120}10000111 \quad \text{and} \quad R_{64} = 0^{59}11011$$

b. Here is the algorithm from the NIST document:

1. Let $L = E(K, 0^b)$.

2. If $MSB_1(L) = 0$, then $K1 = L << 1$;
Else $K_1 = (L << 1) \oplus R_b$;

3. If $MSB_1(K_1) = 0$, then $K_2 = K_1 << 1$;
Else $K_2 = (K_1 << 1) \oplus Rb$.

CHAPTER 13
DIGITAL SIGNATURES AND AUTHENTICATION PROTOCOLS

ANSWERS TO QUESTIONS

13.1 Suppose that John sends an authenticated message to Mary. The following disputes that could arise: **1.** Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share. **2.** John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

13.2 **1.** It must be able to verify the author and the date and time of the signature. **2.** It must be able to authenticate the contents at the time of the signature. **3.** The signature must be verifiable by third parties, to resolve disputes.

13.3 **1.** The signature must be a bit pattern that depends on the message being signed. **2.** The signature must use some information unique to the sender, to prevent both forgery and denial. **3.** It must be relatively easy to produce the digital signature. **4.** It must be relatively easy to recognize and verify the digital signature. **5.** It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message. **6.** It must be practical to retain a copy of the digital signature in storage.

13.4 A **direct digital signature** involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key. An **arbitrated digital signature** operates as follows. Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content. The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter.

13.5 It is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

13.6 **1.** The validity of the scheme depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. **2.** Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

13.7 **Simple replay:** The opponent simply copies a message and replays it later. **Repetition that can be logged:** An opponent can replay a timestamped message within the valid time window. **Repetition that cannot be detected:** This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives. **Backward replay without modification:** This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

13.8 **1.** Attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order. **2.** Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized. **3.** Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

13.9 When a sender's clock is ahead of the intended recipient's clock., an opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the recipient's site. This replay could cause unexpected results.

Answers to Problems

13.1 There are several possible ways to respond to this problem. If public-key encryption is allowed, then of course an arbiter is not needed; A can send message plus signature directly to B. If we constrain the answer to conventional encryption, then the following scenario is possible:

(1) X → A: $M \| \mathrm{E}(K_{xa}, [ID_x \| \mathrm{H}(M)])$

(2) A → Y: $M \| \mathrm{E}(K_{ay}, [ID_x \| \mathrm{H}(M)])$

A can decrypt $M \| E(K_{ay}, [ID_x \| H(M)])$ to determine if M was sent by X.

13.2 The use of a hash function avoids the need for triple encryption.

13.3 X and A, wanting to commit fraud, could disclose PR_x and PR_a, respectively, and claim that these were lost or stolen. The possibility of both private keys becoming public through accident or theft is so unlikely, however, that the sender and arbitrator's claims would have very little credibility.

13.4 It is not so much a protection against an attack as a protection against error. Since N_a is not unique across the network, it is possible for B to mistakenly send message 6 to some other party that would accept N_a.

13.5

(1) A → B: $ID_A \| N_a$

(2) B → KDC: $ID_A \| ID_B \| N_a \| N_b$

(3) KDC → B: $E(PR_{auth}, [ID_A \| PU_a]) \| E(PU_b, E(PR_{auth}, [N_a \| N_b \| K_s \| ID_A \| ID_B]))$

(4) B → A: $E(PU_a, E(PR_{auth}, [N_a \| N_b \| K_s \| ID_A \| ID_B]))$

(5) A → B: $E(K_s, N_b)$

13.6 a. An unintentionally postdated message (message with a clock time that is in the future with respect to the recipient's clock) that requests a key is sent by a client. An adversary blocks this request message from reaching the KDC. The client gets no response and thinks that an omission or performance failure has occurred. Later, when the client is off-line, the adversary replays the suppressed message from the same workstation (with the same network address) and establishes a secure connection in the client's name.

b. An unintentionally postdated message that requests a stock purchase could be suppressed and replayed later, resulting in a stock purchase when the stock price had already changed significantly.

13.7 All three really serve the same purpose. The difference is in the vulnerability. In **Usage 1**, an attacker could breach security by inflating N_a and withholding an answer from B for future replay attack, a form of suppress-replay attack. The attacker could attempt to predict a plausible reply in **Usage 2**, but this will not succeed if the nonces are random. In both Usage 1 and 2, the messages work in either direction. That is, if N is sent in either direction, the response is $E[K, N]$. In **Usage 3**, the message is encrypted in both directions; the purpose of function f is to assure that messages 1 and 2 are not identical. Thus, Usage 3 is more secure.

13.8 Instead of two keys e and d we will have THREE keys u, v, and w. They must be selected in such way that uvw = 1 mod ϕ(N). (This can be done e.g. by selecting u and v randomly (but they have to be prime to ϕ(N)) and then choosing w such that the equation holds.) The key w is made public, while u and v become the first and the second signatory's key respectively. Now the first signatory signs document M by computing $S1 = M^u \bmod N$ The second signatory can verify the signature with the help of his key v and publicly known w, because $S1^{vw} \bmod N$ has to be M. He then 'adds' his signature by computing $S2 = S1^v \bmod N$ (that is $S2 = M^{uv} \bmod N$). Anyone can now verify that S2 is really the double signature of M (i.e. that M was signed by both signatories) because $S2^w \bmod N$ is equal to M only if $S2 = M^{uv} \bmod N$.

13.9 A user who produces a signature with s = 0 is inadvertently revealing his or her private key x via the relationship:

$$s = 0 = k^{-1}[H(m) + xr) \bmod q$$

$$x = \frac{-H(m)}{r} \bmod q$$

13.10 A user's private key is compromised if k is discovered.

13.11 **a.** Note that at the start of step 4, $z = b^{2^j m} \bmod w$. The idea underlying this algorithm is that if ($b^m \bmod w$) ≠ 1 and $w = 1 + 2^a m$ is prime, the sequence of values

$$b^m \bmod w, \quad b^{2m} \bmod w, \quad b^{4m} \bmod w, \ldots$$

will end with 1, and the value just preceding the first appearance of 1 will be $w - 1$. Why? Because, if w is prime, then if we have $z^2 \bmod w = 1$, then we have $z^2 \equiv 1 \bmod w$. And if that is true, then $z = (w - 1)$ or $z = (w + 1)$. We cannot have $z = (w + 1)$, because on the preceding step, z was calculated mod w, so we must have $z = (w - 1)$. On the other hand, if we reach a point where $z = 1$, and z was not equal to $(w - 1)$ on the preceding step, then we know that w is not prime.

b. This algorithm is a simplified version of the Miller-Rabin algorithm. In both cases, a test variable is repeatedly squared and computed modulo the possible prime, and the possible fails if a value of 1 is encountered.

13.12 The signer must be careful to generate the values of k in an unpredictable manner, so that the scheme is not compromised.

13.13 **a.** If Algorithm 1 returns the value g, then we see that $g^q = 1 \pmod p$. Thus, ord(g) divides q. Because q is prime, this implies that ord(g) ∈ {1, q}. However, because $g \neq 1$, we have that ord(g) ≠ 1, and so it must be that ord(g) = q.

b. If Algorithm 2 returns the value g, then we see that $g^q \equiv \left(h^{p-1/q}\right)^q \equiv h^{p-1} \equiv 1 (\mathrm{mod}\, p)$. Thus, ord($g$) divides q. Because q is prime, this implies that ord(g) $\in$ {1, q}. However, because $g \neq 1$, we have that ord(g) $\neq$ 1, and so it must be that ord(g) = q.

c. Algorithm 1 works by choosing elements of Z_p until it finds one of order q. Since q divides p – 1, Z_p contains exactly $\phi(q) = q - 1$ elements of order q. Thus, the probability that $g \in Z_p$ has order q is $(q - 1)/(p - 1)$. When p = 40193 and q = 157 this probability is 156/40192 . So, we expect Algorithm 1 to make 40192/156 ≈ 258 loop iterations.

d. No. If p is 1024 bits and q is 160 bits, then we expect Algorithm 1 to require $(q - 1)/(p - 1) \approx (2^{1024})/(2^{160}) = 2^{864}$ loop iterations.

e. Algorithm 2 will fail to find a generator in its first loop iteration only if $1 \equiv h^{(p-1)/q}$ (mod p). This implies that ord(h) divides $(p - 1)/q$. Thus, the number of bad choices for h is the number of elements of Z_p with order dividing $(p - 1)/q$:

$$\sum_{d|(p-1)/q} \phi(d)$$

This sum is equal to $(p - 1)/q$. Thus, the desired probability is:

$$1 - \frac{(p-1)/q}{p-1} = 1 - \frac{1}{q} = \frac{q-1}{q} = \frac{156}{157} \approx 0.994$$

13.14 a. To verify the signature, the user verifies that $(g^Z)^h = g^X$ mod p.

b. To forge the signature of a message, I find its hash h. Then I calculate Y to satisfy Yh = 1 mod (p-1). Now $g^{Yh} = g$, so $g^{XYh} = g^X$ mod p. Hence (h, g^{XY}) is a valid signature and the opponent can calculate g^{XY} as $(g^X)^Y$.

13.15 a. The receiver validates the digital signature by ensuring that the first 56-bit key in the signature will encipher validation parameter $u1$ into E($k1$, $u1$) if the first bit of M is 0, or that it will encipher $U1$ into E($K1$, $U1$) if the first bit of M is 1; the second 56-bit key in the signature will encipher validation parameter $u2$ into E($k2$, $u2$) if the second bit of M is 0, or it will encipher $U2$ into E($K2$, $U2$) if the second bit of M is 1,; and so on.

b. Only the sender, who knows the private values of ki and Ki and who originally creates vi and Vi from ui and Ui can disclose a key to the receiver. An opponent would have to discover the value of the secret keys from the plaintext-ciphertext pairs of the public key, which was computationally infeasible at the time that 56-bit keys were considered secure.

c. This is a one-time system, because half of the keys are revealed the first time.

d. A separate key must be included in the signature for each bit of the message resulting in a huge digital signature.

CHAPTER 14
AUTHENTICATION APPLICATIONS

ANSWERS TO QUESTIONS

14.1 The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.

14.2 **1.** A user may gain access to a particular workstation and pretend to be another user operating from that workstation. **2.** A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation. **3.** A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

14.3 **1.** Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID). **2.** Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user. **3.** Require the user to prove identity for each service invoked. Also require that servers prove their identity to clients.

14.4 **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link. **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another. **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password. **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

14.5 A full-service Kerberos environment consists of a Kerberos server, a number of clients, and a number of application servers.

14.6 A realm is an environment in which: **1.** The Kerberos server must have the user ID (UID) and hashed password of all participating users in its database. All users are registered with the Kerberos server. **2.** The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

14.7 Version 5 overcomes some environmental shortcomings and some technical deficiencies in Version 4.

14.8 X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

14.9 A chain of certificates consists of a sequence of certificates created by different certification authorities (CAs) in which each successive certificate is a certificate by one CA that certifies the public key of the next CA in the chain.

14.10 The owner of a public-key can issue a certificate revocation list that revokes one or more certificates.

Answers to Problems

14.1 An error in C_1 affects P_1 because the encryption of C_1 is XORed with IV to produce P_1. Both C_1 and P_1 affect P_2, which is the XOR of the encryption of C_2 with the XOR of C_1 and P_1. Beyond that, P_{N-1} is one of the XORed inputs to forming P_N.

14.2 Let us consider the case of the interchange of C_1 and C_2. The argument will be the same for any other adjacent pair of ciphertext blocks. First, if C_1 and C_2 arrive in the proper order:

$$P_1 = E[K, C_1] \oplus IV$$
$$P_2 = E[K, C_2] \oplus C_1 \oplus P_1 = E[K, C_2] \oplus C_1 \oplus E[K, C_1] \oplus IV$$
$$P_3 = E[K, C_3] \oplus C_2 \oplus P_2 = E[K, C_3] \oplus C_2 \oplus E[K, C_2] \oplus C_1 \oplus E[K, C_1] \oplus IV$$

Now suppose that C_1 and C_2 arrive in the reverse order. Let us refer to the decrypted blocks as Q_i.

$$Q_1 = E[K, C_2] \oplus IV$$
$$Q_2 = E[K, C_1] \oplus C_2 \oplus Q_1 = E[K, C_1] \oplus C_2 \oplus E[K, C_2] \oplus IV$$
$$Q_3 = E[K, C_3] \oplus C_1 \oplus Q_2 = E[K, C_3] \oplus C_1 \oplus E[K, C_1] \oplus C_2 \oplus E[K, C_2] \oplus IV$$

The result is that $Q_1 \neq P_1$; $Q_2 \neq P_2$; but $Q_3 = P_3$. Subsequent blocks are clearly unaffected.

14.3 The problem has a simple fix, namely the inclusion of the name of B in the signed information for the third message, so that the third message now reads:

$A \rightarrow B$: $A \{r_B, B\}$

14.4 Taking the *e*th root mod *n* of a ciphertext block will always reveal the plaintext, no matter what the values of *e* and *n* are. In general this is a very difficult problem, and indeed is the reason why RSA is secure. The point is that, if *e* is too small, then taking the normal integer *e*th root will be the same as taking the *e*th root mod *n*, and taking integer *e*th roots is relatively easy.

CHAPTER 15
ELECTRONIC MAIL SECURITY

ANSWERS TO QUESTIONS

15.1 Authentication, confidentiality, compression, e-mail compatibility, and segmentation

15.2 A detached signature is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent virus infection. Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on.

15.3 **a.** It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required. **b.** Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

15.4 R64 converts a raw 8-bit binary stream to a stream of printable ASCII characters. Each group of three octets of binary data is mapped into four ASCII characters.

15.5 When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text.

15.6 E-mail facilities often are restricted to a maximum message length.

15.7 PGP includes a facility for assigning a level of trust to individual signers and to keys.

15.8 RFC 822 defines a format for text messages that are sent using electronic mail.

15.9 MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail.

15.10 S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security.

Answers to Problems

15.1 CFB avoids the need to add and strip padding.

15.2 This is just another form of the birthday paradox discussed in Appendix 11A. Let us state the problem as one of determining what number of session keys must be generated so that the probability of a duplicate is greater than 0.5. From Equation (11.6) in Appendix 11A, we have the approximation:

$$k = 1.18 \times \sqrt{n}$$

For a 128-bit key, there are 2^{128} possible keys. Therefore

$$k = 1.18 \times \sqrt{2^{128}} = 1.18 \times 2^{64}$$

15.3 Again, we are dealing with a birthday-paradox phenomenon. We need to calculate the value for:

P(n, k) = Pr [at least one duplicate in k items, with each item able to take on one of n equally likely values between 1 and n]

In this case, k = N and n = 2^{64}. Using equation (11.5) of Appendix 1A:

$$\mathrm{P}\left(2^{64}, N\right) = 1 - \frac{2^{64}!}{\left(2^{64} - N\right)! 2^{64 \times k}}$$
$$> 1 - \mathrm{e}^{-[N \times (N-1)]/2^{65}}$$

15.4 **a.** Not at all. The message digest is encrypted with the sender's private key. Therefore, anyone in possession of the public key can decrypt it and recover the entire message digest.

b. The probability that a message digest decrypted with the wrong key would have an exact match in the first 16 bits with the original message digest is 2^{-16}.

15.5 We trust this owner, but that does not necessarily mean that we can trust that we are in possession of that owner's public key.

15.6 It certainly provides more security than a monoalphabetic substitution. Because we are treating the plaintext as a string of bits and encrypting 6 bits at a time, we are not encrypting individual characters. Therefore, the frequency information is lost, or at least significantly obscured.

15.7 DES is unsuitable because of its short key size. Two-key triple DES, which has a key length of 112 bits, is suitable. AES is also suitable.

Chapter 16
IP Security

Answers to Questions

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

16.2 Access control; connectionless integrity; data origin authentication; rejection of replayed packets (a form of partial sequence integrity); confidentiality (encryption); and limited traffic flow confidentiality

16.3 A security association is uniquely identified by three parameters: **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed. **IP Destination Address:** Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router. **Security Protocol Identifier:** This indicates whether the association is an AH or ESP security association.

A security association is normally defined by the following parameters: **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers, described in Section 16.3 (required for all implementations). **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations). **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay, described in Section 16.3 (required for all implementations). **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations).

ESP Information: Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations). **Lifetime of this Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations). **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section. **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

16.4 **Transport mode** provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. **Tunnel mode** provides protection to the entire IP packet.

16.5 A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits it to the intended destination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence.

16.6 **1.** If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is used to expand the plaintext (consisting of the Payload Data, Padding, Pad Length, and Next Header fields) to the required length. **2.** The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment. **3.** Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

16.7 **Transport adjacency:** Refers to applying more than one security protocol to the same IP packet, without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one IPSec instance: the (ultimate) destination. **Iterated tunneling:** Refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPSec site along the path.

16.8 ISAKMP by itself does not dictate a specific key exchange algorithm; rather, ISAKMP consists of a set of message types that enable the use of a variety of key exchange algorithms. Oakley is the specific key exchange algorithm mandated for use with the initial version of ISAKMP.

ANSWERS TO PROBLEMS

16.1 a. Immutable: Version, Internet Header Length, Total Length, Identification, Protocol (This should be the value for AH.), Source Address, Destination Address (without loose or strict source routing). None of these are changed by routers in transit.

Mutable but predictable: Destination Address (with loose or strict source routing). At each intermediate router designated in the source routing list, the Destination Address field is changed to indicate the next designated address. However, the source routing field contains the information needed for doing the MAC calculation.

Mutable (zeroed prior to ICV calculation): Type of Service (TOS), Flags, Fragment Offset, Time to Live (TTL), Header Checksum. TOS may be altered by a router to reflect a reduced service. Flags and Fragment offset are altered if an router performs fragmentation. TTL is decreased at each router. The Header Checksum changes if any of these other fields change.

b. Immutable: Version, Payload Length, Next Header (This should be the value for AH.), Source Address, Destination Address (without Routing Extension Header)

Mutable but predictable: Destination Address (with Routing Extension Header)

Mutable (zeroed prior to ICV calculation): Class, Flow Label, Hop Limit

c. IPv6 options in the Hop-by-Hop and Destination Extension Headers contain a bit that indicates whether the option might change (unpredictably) during transit.

Mutable but predictable: Routing

Not Applicable: Fragmentation occurs after outbound IPSec processing and reassembly occur before inbound IPSec processing , so the Fragmentation Extension Header, if it exists, is not seen by IPSec.

16.2 From RFC 2401

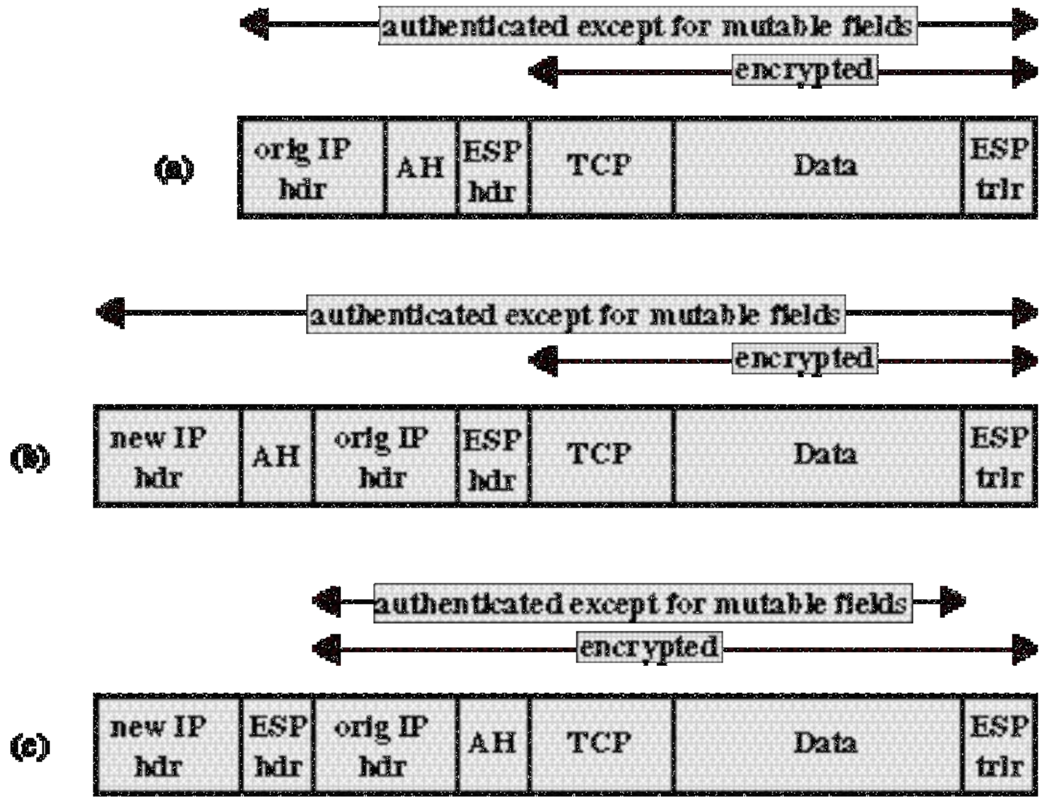
IPv4 Header Fields	Outer Header at Encapsulator	Inner Header at Decapsulator
version	4 (1)	no change
header length	constructed	no change
TOS	copied from inner header (5)	no change
total length	constructed	no change
ID	constructed	no change
Flags	constructed, DF (4)	no change
Fragment offset	constructed	no change
TTL	constructed	decrement (2)
protocol	AH, ESP, routing header	no change

checksum	constructed	no change
source address	constructed (3)	no change
destination address	constructed (3)	no change
options	never copied	no change

IPv6 Header Fields	Outer Header at Encapsulator	Inner Header at Decapsulator
version	6 (1)	no change
class	copied or configured (6)	no change
flow id	copied or configured	no change
length	constructed	no change
next header	AH, ESP, routing header	no change
hop count	constructed (2)	decrement (2)
source address	constructed (3)	no change
dest address	constructed (3)	no change
extension headers	never copied	no change

1. The IP version in the encapsulating header can be different from the value in the inner header.
2. The TTL in the inner header is decremented by the encapsulator prior to forwarding and by the decapsulator if it forwards the packet.
3. src and dest addresses depend on the SA, which is used to determine the dest address, which in turn determines which src address (net interface) is used to forward the packet.
4. configuration determines whether to copy from the inner header (IPv4 only), clear or set the DF.
5. If Inner Hdr is IPv4, copy the TOS. If Inner Hdr is IPv6, map the Class to TOS.
6. If Inner Hdr is IPv6, copy the Class. If Inner Hdr IPv4, map the TOS to Class.

16.3 We show the results for IPv4; IPv6 is similar.



16.4 This order of processing facilitates rapid detection and rejection of replayed or bogus packets by the receiver, prior to decrypting the packet, hence potentially reducing the impact of denial of service attacks. It also allows for the possibility of parallel processing of packets at the receiver, i.e., decryption can take place in parallel with authentication.

16.5 **a.** The Aggressive Exchange type.

b. $(CKY_I, CKY_R) \leftrightarrow$ HDR

(OK_KEYX) $\leftrightarrow$ HDR

(GRP) $\leftrightarrow$ P

$g^x, g^y) \leftrightarrow$ KE

(EHAO, EHAS) $\leftrightarrow$ T

(NIDP) $\leftrightarrow$ HDR

$(ID_I, ID_R) \leftrightarrow$ ID

$(N_I, N_R) \leftrightarrow$ NONCE

$(S_{KI}[X], S_{KR}[X]) \leftrightarrow$ SIG

CHAPTER 17
WEB SECURITY

ANSWERS TO QUESTIONS

17.1 The advantage of using **IPSec** (Figure 17.1a) is that it is transparent to end users and applications and provides a general-purpose solution. Further, IPSec includes a filtering capability so that only selected traffic need incur the overhead of IPSec processing. The advantage of using **SSL** is that it makes use of the reliability and flow control mechanisms of TCP. The advantage **application-specific security services** (Figure 17.1c) is that the service can be tailored to the specific needs of a given application.

17.2 SSL handshake protocol; SSL change cipher spec protocol; SSL alert protocol; SSL record protocol.

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

17.4 **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state. **Peer certificate:** An X509.v3 certificate of the peer. **Compression method:** The algorithm used to compress data prior to encryption. **Cipher spec:** Specifies the bulk data encryption algorithm (such as null, DES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size. **Master secret:** 48-byte secret shared between the client and server. **Is resumable:** A flag indicating whether the session can be used to initiate new connections.

17.5 **Server and client random:** Byte sequences that are chosen by the server and client for each connection. **Server write MAC secret:** The secret key used in MAC operations on data sent by the server. **Client write MAC secret:** The secret key used in MAC operations on data sent by the client. **Server write key:** The conventional encryption key for data encrypted by the server and decrypted by the client. **Client write key:** The conventional encryption key for data encrypted by the client and decrypted by the server. **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record. **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed $2^{64} - 1$.

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

17.7 Fragmentation; compression; add MAC; encrypt; append SSL record header.

17.8 **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer. **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a Web site or by electronic mail. A merchant that accepts payment cards must have a relationship with an acquirer. **Issuer:** This is a financial institution, such as a bank, that provides the cardholder with the payment card. Typically, accounts are applied for and opened by mail or in person. Ultimately, it is the issuer that is responsible for the payment of the debt of the cardholder. **Acquirer:** This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers. The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. Subsequently, the acquirer is reimbursed by the issuer over some sort of payment network for electronic funds transfer. **Payment gateway:** This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system. **Certification authority (CA):** This is an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose. As was discussed in previous chapters, a hierarchy of CAs is used, so that participants need not be directly certified by a root authority.

17.9 A dual signature is used to sign two concatenated documents each with its own hash code. The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer want to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order.

Answers to Problems

17.1 The change cipher spec protocol exists to signal transitions in ciphering strategies, and can be sent independent of the complete handshake protocol exchange.

17.2 **a. Brute Force Cryptanalytic Attack:** The conventional encryption algorithms use key lengths ranging from 40 to 168 bits.

b. Known Plaintext Dictionary Attack: SSL protects against this attack by not really using a 40-bit key, but an effective key of 128 bits. The rest of the key is constructed from data that is disclosed in the Hello messages. As a result the dictionary must be long enough to accommodate 2^{128} entries.

c. Replay Attack: This is prevented by the use of nonces..

d. Man-in-the-Middle Attack: This is prevented by the use of pubic-key certificates to authenticate the correspondents.

e. Password Sniffing: User data is encrypted.

f. IP Spoofing: The spoofer must be in possession of the secret key as well as the forged IP address..

g. IP Hijacking: Again, encryption protects against this attack..

h. SYN Flooding: SSL provides no protection against this attack.

17.3 SSL relies on an underlying reliable protocol to assure that bytes are not lost or inserted. There was some discussion of reengineering the future TLS protocol to work over datagram protocols such as UDP, however, most people at a recent TLS meeting felt that this was inappropriate layering (from the SSL FAQ).

CHAPTER 18
INTRUDERS

ANSWERS TO QUESTIONS

18.1 **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account. **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges. **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

18.2 **One-way encryption:** The system stores only an encrypted form of the user's password. When the user presents a password, the system encrypts that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed-length output is produced. **Access control:** Access to the password file is limited to one or a very few accounts.

18.3 **1.** If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved. **2.** An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions. **3.** Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

18.4 **Statistical anomaly detection** involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior. **Rule-Based Detection** involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

18.5 **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. **Interval timer:** The length of time between two related events. **Resource utilization:** Quantity of resources consumed during a specified period.

18.6 With **rule-based anomaly detection**, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior. **Rule-based penetration identification** uses rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules used in these systems are specific to the machine and operating system. Also, such rules are generated by "experts" rather than by means of an automated analysis of audit records.

18.7 Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems.

18.8 The salt is combined with the password at the input to the one-way encryption routine.

18.9 **User education:** Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. **Computer-generated passwords:** Users are provided passwords generated by a computer algorithm. **Reactive password checking:** the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. **Proactive password checking:** a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it.

Answers to Problems

18.1 Let WB equal the event {witness reports Blue cab}. Then:

$$\Pr[\text{Blue/WB}] = \frac{\Pr[\text{WB/Blue}]\Pr[\text{Blue}]}{\Pr[\text{WB/Blue}]\Pr[\text{Blue}] + \Pr[\text{WB/Green}]\Pr[\text{Green}]}$$

$$= \frac{(0.8)(0.15)}{(0.8)(0.15) + (0.2)(0.85)} = 0.41$$

This example, or something similar, is referred to as "the juror's fallacy."

18.2 **a.** T = $\frac{26^4}{2}$ seconds = 63.5 hours

b. Expect 13 tries for each digit. T = 13 × 4 = 52 seconds.

18.3 **a.** $p = r^k$

b. $p = \frac{r^k - r^p}{r^{k+p}}$

c. $p = r^p$

18.4 **a.** T = $(21 \times 5 \times 21)^2$ = 4,862,025

b. p = 1/T ≈ 2×10^{-7}

18.5 There are $95^{10} \approx 6 \times 10^{19}$ possible passwords. The time required is:

$$\frac{6 \times 10^{19} \text{passwords}}{6.4 \times 10^{6} \text{passwords/ second}} = 9.4 \times 10^{12} \text{seconds}$$

$$= 300,000 \text{ years}$$

18.6 **a.** Since PU_a and PR_a are inverses, the value PR_a can be checked to validate that P_a was correctly supplied: Simply take some arbitrary block X and verify that X = D(PRa, E[PUa, X]).

b. Since the file /etc/publickey is publicly readable, an attacker can guess P (say P') and compute $PR_{a'}$ = D(P', E[P, PR_a]). now he can choose an arbitrary block Y and check to see if Y = D(PR_a, E[PU_a, Y]). If so, it is highly probable that P' = P. Additional blocks can be used to verify the equality.

18.7 Yes.

18.8 Without the salt, the attacker can guess a password and encrypt it. If ANY of the users on a system use that password, then there will be a match. With the salt, the attacker must guess a password and then encrypt it once for each user, using the particular salt for each user.

18.9 It depends on the size of the user population, not the size of the salt, since the attacker presumably has access to the salt for each user. The benefit of larger salts is that the larger the salt, the less likely it is that two users will have the same salt. If multiple users have the same salt, then the attacker can do one encryption per password guess to test all of those users.

18.10 **a.** If there is only one hash function (k = 1), which produces one of N possible hash values, and there is only one word in the dictionary, then the probability that an arbitrary bit b_i is set to 1 is just 1/N. If there are k hash functions, let us assume for simplicity that they produce k distinct hash functions for a given word. This assumption only introduces a small margin of error. Then, the probability that an arbitrary bit b_i is set to 1 is k/N. Therefore, the probability that b_i is equal to 0 is 1 – k/N. The probability that a bit is left unset after D dictionary words are processed is just the probability that each of the D transformations set other bits:

$$\Pr[b_i = 0] = \left(1 - \frac{k}{N}\right)^D$$

This can also be interpreted as the expected fraction of bits that are equal to 0.

b. A word not in the dictionary will be falsely accepted if all k bits tested are equal to 1. Now, from part (a), we can say that the expected fraction of bits in the hash table that are equal to one is $1 - \phi$. The probability that a random word will be mapped by a single hash function onto a bit that is already set is the probability that the bit generated by the hash function is in the set of bits equal to one, which is just $1 - \phi$. Therefore, the probability that the k hash functions applied to the word will produce k bits all of which are in the set of bits equal to one is $(1 - \phi)^k$.

c. We use the approximation $(1 - x) \approx e^{-x}$.

18.11 The system enciphers files with a master system key KM, which is stored in some secure fashion. When User i attempts to read file F, the header of F is decrypted using KM and User i's read privilege is checked. If the user has read access, the file is decrypted using KM and the reencrypted using User i's key for transmission to User i. Write is handled in a similar fashion.

CHAPTER 19
MALICIOUS SOFTWARE

ANSWERS TO QUESTIONS

19.1 A virus may use compression so that the infected program is exactly the same length as an uninfected version.

19.2 A portion of the virus, generally called a *mutation engine*, creates a random encryption key to encrypt the remainder of the virus. The key is stored with the virus, and the mutation engine itself is altered. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.

19.3 A dormant phase, a propagation phase, a triggering phase, and an execution phase

19.4 **1.** Search for other systems to infect by examining host tables or similar repositories of remote system addresses. **2.**Establish a connection with a remote system. **3.** Copy itself to the remote system and cause the copy to be run.

19.5 This system provides a general-purpose emulation and virus-detection system. The objective is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced. When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running a general antivirus program so that it can be detected before it is allowed to run elsewhere.

19.6 Behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. The behavior blocking software then blocks potentially malicious actions before they have a chance to affect the system.

19.7 A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.

ANSWERS TO PROBLEMS

19.1 The program will loop indefinitely once all of the executable files in the system are infected.

19.2 D is supposed to examine a program P and return TRUE if P is a computer virus and FALSE if it is not. But CV calls D. If D says that CV is a virus, then CV will not infect an executable. But if D says that CV is not a virus, it infects an executable. D always returns the wrong answer.

CHAPTER 20
FIREWALLS

ANSWERS TO QUESTIONS

20.1 **1.** All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible, as explained later in this section. **2.** Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies, as explained later in this section. **3.** The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system.

20.2 **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service. **Direction control:** Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall. **User control:** Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology, such as is provided in IPSec. **Behavior control:** Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

20.3 **Source IP address:** The IP address of the system that originated the IP packet. **Destination IP address:** The IP address of the system the IP packet is trying to reach. **Source and destination transport-level address:** The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET. **IP protocol field:** Defines the transport protocol. **Interface:** For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for.

20.4 **1.** Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application-specific vulnerabilities or functions. For example, a packet filter firewall cannot block specific application commands; if a packet filter firewall allows a given application, all functions available within that application will be permitted. **2.** Because of the limited information available to the firewall, the logging functionality present in packet filter firewalls is limited. Packet filter logs normally contain the same information used to make access control decisions (source address, destination address, and traffic type). **3.** Most packet filter firewalls do not support advanced user authentication schemes. Once again, this limitation is mostly due to the lack of upper-layer functionality by the firewall. **4.** They are generally vulnerable to attacks and exploits that take advantage of problems within the TCP/IP specification and protocol stack, such as *network layer address spoofing*. Many packet filter firewalls cannot detect a network packet in which the OSI Layer 3 addressing information has been altered. Spoofing attacks are generally employed by intruders to bypass the security controls implemented in a firewall platform. **5.** Finally, due to the small number of variables used in access control decisions, packet filter firewalls are susceptible to security breaches caused by improper configurations. In other words, it is easy to accidentally configure a packet filter firewall to allow traffic types, sources, and destinations that should be denied based on an organization's information security policy.

20.5 A **traditional packet filter** makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. A **stateful inspection packet filter** tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, as shown in Table 20.2. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory

20.6 An application-level gateway, also called a proxy server, acts as a relay of application-level traffic.

20.7 A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

20.8 The **screened host firewall, single-homed bastion** configuration (Figure 20.2a), the firewall consists of two systems: a packet-filtering router and a bastion host; the latter performs authentication and proxy functions. In the single-homed configuration just described, if the packet-filtering router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network. The **screened host firewall, dual-homed bastion** configuration physically prevents such a security breach. In the **screened subnet firewall** configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability.

20.9 A **subject** is an entity capable of accessing objects. Generally, the concept of subject equates with that of process. Any user or application actually gains access to an object by means of a process that represents that user or application. An **object** is anything to which access is controlled. Examples include files, portions of files, programs, and segments of memory.

20.10 For each object, an **access control list** lists users and their permitted access rights. A **capability ticket** specifies authorized objects and operations for a user.

20.11 **No read up:** A subject can only read an object of less or equal security level. **No write down:** A subject can only write into an object of greater or equal security level.

20.12 **Complete mediation:** The security rules are enforced on every access, not just, for example, when a file is opened. **Isolation:** The reference monitor and database are protected from unauthorized modification. **Verifiability:** The reference monitor's correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation.

20.13 The Common Criteria (CC) for Information Technology and Security Evaluation is an international initiative by standards bodies in a number of countries to develop international standards for specifying security requirements and defining evaluation criteria.

Answers to Problems

20.1 It will be impossible for the destination host to complete reassembly of the packet if the first fragment is missing, and therefore the entire packet will be discarded by the destination after a time-out.

20.2 When a TCP packet is fragmented so as to force interesting header fields out of the zero-offset fragment, there must exist a fragment with FO equal to 1. If a packet with FO = 1 is seen, conversely, it could indicate the presence, in the fragment set, of a zero-offset fragment with a transport header length of eight octets Discarding this one-offset fragment will block reassembly at the receiving host and be as effective as the direct method described above.

20.3 If the router's filtering module enforces a minimum fragment offset for fragments that have non-zero offsets, it can prevent overlaps in filter parameter regions of the transport headers.

20.4 The purpose of the "no write down" rule, or *-property is to address the problem of Trojan horse software. With the *-property, information cannot be compromised through the use of a Trojan horse. Under this property, a program operating on behalf of one user cannot be used to pass information to any user having a lower or disjoint access class.

20.5 Drake is not authorized to read the string directly, so the no-read-up rule will prevent this. Similarly, Drake is not authorized to assign a security level of sensitive to the back-pocket file, so that is prevented as well.