SOLUTIONS MANUAL

CRYPTOGRAPHY AND NETWORK SECURITY: PRINCIPLES AND PRACTICE EIGHTH EDITION

CHAPTERS 1–10

WILLIAM STALLINGS

TABLE OF CONTENTS

NOTICE

This manual contains solutions to the review questions and homework problems in *Cryptography and Network Security, Eighth 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 wllmst@me.net. An errata sheet for this manual, if needed, is available at https://www.box.com/shared/nh8hti5167 File name is S-Crypto8e-mmyy.

W.S.

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

1.3 **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).

1.4 **Cryptographic algorithms**: Transform data between plaintext and ciphertext.
Data integrity: Mechanisms used to assure the integrity of a data unit or stream of data units.
Digital signature: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery.

Authentication exchange: A mechanism intended to ensure the identity of an entity by means of information exchange.
Traffic padding: The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
Routing control: Enables selection of particular physically or logically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
Notarization: The use of a trusted third party to assure certain properties of a data exchange.
Access control: A variety of mechanisms that enforce access rights to resources.

1.5 **Keyless**: Do not use any keys during cryptographic transformations.
Single-key: The result of a transformation are a function of the input data and a single key, known as a secret key.
Two-key: At various stages of the calculate two different but related keys are used, referred to as private key and public key.

1.6 **Communications security**: Deals with the protection of communications through the network, including measures to protect against both passive and active attacks.
Device security: Deals with the protection of network devices, such as routers and switches, and end systems connected to the network, such as client systems and servers.

1.7 **Trust**: The willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party.
Trustworthiness: A characteristic of an entity that reflects the degree to which that entity is deserving of trust.

ANSWERS TO PROBLEMS

1.1 The system must keep personal identification numbers confidential, both in the host system and during transmission for a transaction. It must protect the integrity of account records and of individual transactions. Availability of the host system is important to the economic well being of the bank, but not to its fiduciary responsibility. The availability of individual teller machines is of less concern.

1.2 The system does not have high requirements for integrity on individual transactions, as lasting damage will not be incurred by occasionally losing a call or billing record. The integrity of control programs and configuration records, however, is critical. Without these, the switching function would be defeated and the most important attribute of all - availability - would be compromised. A telephone switching system must also preserve the confidentiality of individual calls, preventing one caller from overhearing another.

1.3a. The system will have to assure confidentiality if it is being used to publish corporate proprietary material.

b. The system will have to assure integrity if it is being used to laws or regulations.

c. The system will have to assure availability if it is being used to publish a daily paper.

1.4a. An organization managing public information on its web server determines that there is no potential impact from a loss of confidentiality (i.e., confidentiality requirements are not applicable), a moderate potential impact from a loss of integrity, and a moderate potential impact from a loss of availability.

b. A law enforcement organization managing extremely sensitive investigative information determines that the potential impact from a loss of confidentiality is high, the potential impact from a loss of integrity is moderate, and the potential impact from a loss of availability is moderate.

c. A financial organization managing routine administrative information (not privacy-related information) determines that the potential impact from a loss of confidentiality is low, the potential impact from a loss of integrity is low, and the potential impact from a loss of availability is low.

d. The management within the contracting organization determines that: (i) for the sensitive contract information, the potential impact from a loss of confidentiality is moderate, the potential impact from a loss of integrity is moderate, and the potential impact from a loss of availability is low; and (ii) for the routine administrative information (non-privacy-related information), the potential impact from a loss of confidentiality is low, the potential impact from a loss of integrity is low, and the potential impact from a loss of availability is low.

e. The management at the power plant determines that: (i) for the sensor data being acquired by the SCADA system, there is no potential impact from a loss of confidentiality, a high potential impact from a loss of integrity, and a high potential impact from a loss of availability; and (ii) for the administrative information being processed by the system, there is a low potential impact from a loss of confidentiality, a low potential impact from a loss of integrity, and a low potential impact from a loss of availability. (Examples from FIPS 199.)

Chapter 2 Introduction to Number Theory

Answers to Questions

2.1 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.

2.2 It means that *b* is a divisor of *a*.

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

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

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

2.6 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.

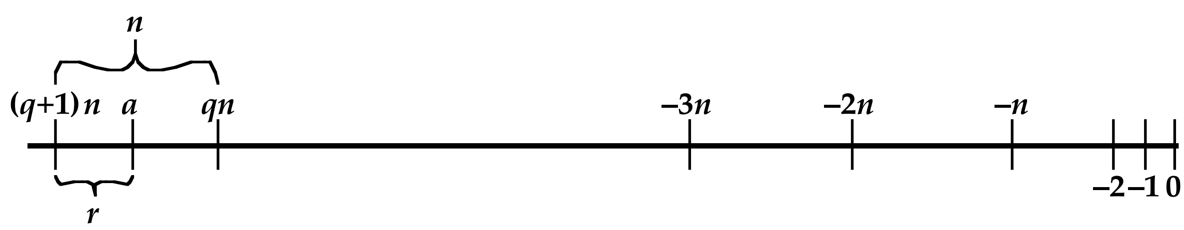
2.7 If *r* and *n* are relatively prime integers with $n > 0$. and if $\phi(n)$ is the least positive exponent *m* such that $a^m \equiv 1 \bmod n$, then *r* is called a primitive root modulo *n*.

2.8 The two terms are synonymous.

Answers to Problems

2.1 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 $0 \le r \le n$.

2.2 In this diagram, q is a negative integer.



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

2.4 Section 2.3 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. $-5 \bmod -3 = -5 - (-3) \lfloor (-5)/(-3) \rfloor = -2$

2.5 $a = b$

2.6 Recall Figure 2.1 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.

2.7 1, 2, 4, 6, 16, 12

2.8 **a.** This is the definition of congruence as used in Section 2.3.
b. The first two statements mean

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

so that

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

2.9 **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

2.10 $1^{-1} = 1$, $2^{-1} = 3$, $3^{-1} = 2$, $4^{-1} = 4$

2.11 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 $N \equiv a_0 + a_1 + \ldots + a_{n-1} \pmod 9$.

2.12 **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

2.13 **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

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

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 $1 < 2j + 1 \le k + 2$, 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.

2.14 **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.

2.15 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, $C_n \times \gcd(A_n, B_n) = C_n \times A_n = \gcd(A, B)$.

2.16 a. 3239

b. gcd(40902, 24240) = 34 ≠ 1, so there is no multiplicative inverse.
c. 550

2.17 **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 $X - p_1 p_2 \ldots p_n = 1$, 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 $X = 1 + p_1 p_2 \ldots p_n$, so p_{n+1} is equal to or less than this prime. Therefore, since this prime divides X, it is ≤ X and therefore $p_{n+1} \le X$.

2.18 **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

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

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

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

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

2.20 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

$$3^{201} = (3^{10})^{20} \times 3 \equiv 3 \pmod{11}.$$

2.21 12

2.22 6

2.23 1

2.24 6

2.25 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.

2.26 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 $\phi(p^n) = p^n - p^{n-1}$.

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

2.28 It follows immediately from the result stated in Problem 2.26.

2.29 totient

2.30 **a.** For n = 5, $2^n - 2 = 30$, which is divisible by 5.
b. We can rewrite the Chinese test as $(2^n - 2) \equiv 0$ mod n, or equivalently,
$2^n \equiv 2$ (mod n). By Fermat's Theorem, this relationship is true **if** n is prime (Equation 2.10).
c. For n = 15, $2^n - 2 = 32{,}766$, which is divisible by 15.
d. $2^{10} = 1024 \equiv 1$ (mod 341)
$2^{340} = (2^{10})^{34} \equiv$ (1 mod 341)
$2^{341} \equiv 2$ (mod 341)

2.31 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.

2.32 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.

2.33 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

$$N_iM_i \equiv 1 \pmod{m_i}$$

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

$$x \equiv a_1N_1M_1 + \ldots + a_kN_kM_k \pmod{M}$$

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

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

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 $N_iM_i \equiv 1 \pmod{m_i}$, the solution is also unique modulo M, which proves this form of the Chinese Remainder Theorem.

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

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

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

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

2.35 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:

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

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

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.

2.36 2, 3, 8, 12, 13, 17, 22, 23

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

Chapter 3 Classical Encryption Techniques

Answers to Questions

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

3.2 Permutation and substitution.

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

3.4 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.

3.5 Cryptanalysis and brute force.

3.6 **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.

3.7 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.

3.8 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.

3.9 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.

3.10 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.

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

3.12 **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.

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

ANSWERS TO PROBLEMS

3.1 **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 $q = p + m/k \ne p$.

3.2 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 $12 \times 26 = 312$.

3.3 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 $1 = (12 + b) \bmod 26$. By observation, $b = 15$.

3.4 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)

3.5 **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 monoalphabetic 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.

3.6 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)

3.7 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.

3.8 SPUTNIK

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

3.10 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

3.11 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 3.10a, by rotating the columns by one step and the rows by three steps.

3.12 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!

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

3.14 **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 $(9 \times 7) - (4 \times 5) = 43$. 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.

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

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

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

$$(B, A, A, \ldots, A, A) \leftrightarrow K_1$$
$$(A, B, A, \ldots, A, A) \leftrightarrow K_2$$
$$\vdots$$
$$(A, A, A, \ldots, A, B) \leftrightarrow K_n$$

3.16 **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

3.17 **a.** $(80 - 10) \bmod 26 = 18$
b. $[(1 \times 9 \times 5) + (7 \times 2 \times 1) + (22 \times 4 \times 2) - (22 \times 9 \times 1) - (2 \times 2 \times 1) - (5 \times 7 \times 4)] \bmod 26$
$= (45 + 14 + 176 - 198 - 4 - 140) \bmod 26$
$= (-107) \bmod 26 = 23$

3.18 We label the matrices as **A** and **B**, respectively.
a. $\det(\mathbf{A}) = (44 - 3) \bmod 26 = 15$
$(\det(\mathbf{A}))^{-1} = 7$, using Table E.1 of Appendix E

$$\mathbf{A}^{-1} = (\det(\mathbf{A}))^{-1} \begin{pmatrix} \mathrm{cof}_{11}(\mathbf{A}) & \mathrm{cof}_{21}(\mathbf{A}) \\ \mathrm{cof}_{12}(\mathbf{A}) & \mathrm{cof}_{22}(\mathbf{A}) \end{pmatrix} =$$

$$7 \times \begin{pmatrix} 22 & -3 \\ -1 & 2 \end{pmatrix} \bmod 26 = \begin{pmatrix} 154 & -21 \\ -7 & 14 \end{pmatrix} \bmod 26 = \begin{pmatrix} 24 & 5 \\ 19 & 14 \end{pmatrix}$$

b. det(**B**) =[(6 × 16 × 15) + (24 × 10 × 20) + (1 × 13 × 17) – (1 × 16 × 20) – (10 × 17 × 6) – (15 × 24 × 13)] mod 26
= (1440 + 4800 + 221 – 320 – 1020 – 4680) mod 26
= 441 mod 26 = 25
We use the formulas from Appendix E

$$b_{ij} = \frac{\mathrm{cof}_{ji}(\mathbf{K})}{\det(\mathbf{K})} \bmod 26 = 17 \times \mathrm{cof}_{ji}(\mathbf{K}) \bmod 26$$

$$b_{11} = \begin{vmatrix} 16 & 10 \\ 17 & 15 \end{vmatrix} \times 25 \bmod 26 = (16 \times 15 - 10 \times 17) \times 25 \bmod 26 = 5100 \bmod 26 = 8$$

$$b_{12} = -\begin{vmatrix} 24 & 1 \\ 17 & 15 \end{vmatrix} \times 25 \bmod 26 = -(24 \times 15 - 1 \times 17) \times 25 \bmod 26 = -8575 \bmod 26 = 5$$

$$b_{13} = \begin{vmatrix} 24 & 1 \\ 16 & 10 \end{vmatrix} \times 25 \bmod 26 = (24 \times 10 - 1 \times 16) \times 25 \bmod 26 = 5600 \bmod 26 = 10$$

$$b_{21} = -\begin{vmatrix} 13 & 10 \\ 20 & 15 \end{vmatrix} \times 25 \bmod 26 = -(13 \times 15 - 10 \times 20) \times 25 \bmod 26 = 125 \bmod 26 = 21$$

$$b_{22} = \begin{vmatrix} 6 & 1 \\ 20 & 15 \end{vmatrix} \times 25 \bmod 26 = (6 \times 15 - 1 \times 20) \times 25 \bmod 26 = 1750 \bmod 26 = 8$$

$$b_{23} = -\begin{vmatrix} 6 & 1 \\ 13 & 10 \end{vmatrix} \times 25 \bmod 26 = -(6 \times 10 - 1 \times 13) \times 25 \bmod 26 = -1175 \bmod 26 = 21$$

$$b_{31} = \begin{vmatrix} 13 & 16 \\ 20 & 17 \end{vmatrix} \times 25 \bmod 26 = (13 \times 17 - 16 \times 20) \times 25 \bmod 26 = -2475 \bmod 26 = 21$$

$$b_{32} = -\begin{vmatrix} 6 & 24 \\ 20 & 17 \end{vmatrix} \times 25 \bmod 26 = -(6 \times 17 - 24 \times 20) \times 25 \bmod 26 = 9450 \bmod 26 = 12$$

$$b_{33} = \begin{vmatrix} 6 & 24 \\ 13 & 16 \end{vmatrix} \times 25 \bmod 26 = (6 \times 16 - 24 \times 13) \times 25 \bmod 26 = -5400 \bmod 26 = 8$$

$$\mathbf{B}^{-1} = \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix}$$

3.19
```
key:         legleglegle
plaintext:   explanation
ciphertext:  PBVWETLXOZR
```

3.20 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

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

3.22 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 4 Block Ciphers and the Data Encryption Standard

Answers to Questions

4.1 Many 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.

4.2 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.

4.3 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.

4.4 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.

4.5 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.

4.6 **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.

4.7 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.

ANSWERS TO PROBLEMS

4.1 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 4.1. In this case the key size is $n \times 2n$ and the huge table is not required.

4.2 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*.

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

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

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

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

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

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

4.4 **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 $E(K, \bullet)$ and $E(K', \bullet)$. 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 4.3 we have that

Pr(t' 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 $t' = N - t$.

4.5 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 $c \in \{0, 1\}^{128}$.

4.6 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.

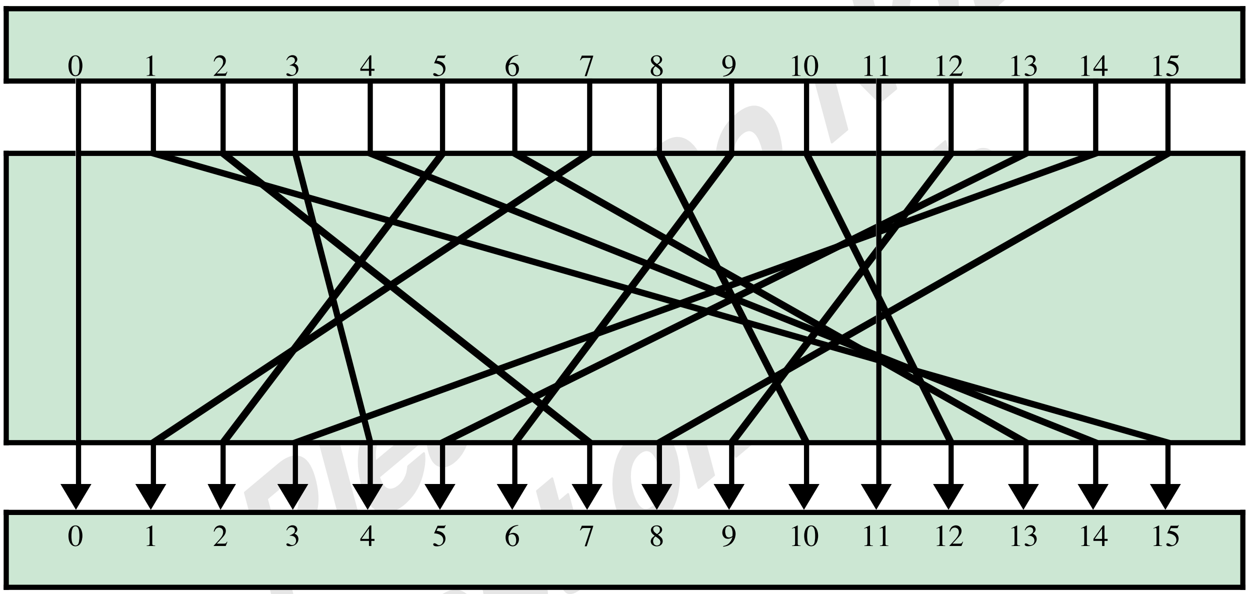
4.7 The reasoning for the Feistel cipher, as shown in Figure 4.3, applies in the case of DES. We only have to show the effect of the IP and IP^{-1} functions. For encryption, the input to the final IP^{-1} is $RE_{16} \,||\, LE_{16}$. The output of that stage is the ciphertext. On decryption, the first 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.

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

4.9



4.10 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,

$E(RD_0) \oplus K_{16}$ = 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'.

4.11 a. First, pass the 64-bit input through PC-1 (Table 4.4a) 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 4.4b) 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 4.2a):

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 4.2c) 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 4.2d, 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.

4.12 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.

4.13

Round number	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1
Bits rotated	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

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

A	B	A ⊕ B	(A ⊕ B)'	A' ⊕ 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 ⊕ B = A' ⊕ B', which is easily seen to be true. Consider the two XOR operations in Figure 4.6. 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.

4.15 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.

CHAPTER 5 FINITE FIELDS

ANSWERS TO QUESTIONS

5.1 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.

5.2 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.

5.3 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.

5.4 **(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

5.1 **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}.

5.2 Here are the addition and multiplication tables

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

×	0	1	2
0	0	0	0
1	0	1	2
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.

5.3 S is a ring. We show using the axioms in Figure 5.2:

(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.

5.4

+	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

5.5 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 5.2, 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.

5.6 **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 ≠ 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).

5.7 **a.** $9x^2 + 7x + 7$

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

5.8 **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: $(x + 1)^4$

5.9 **a.** 1

b. 1

c. x + 1

d. x + 78

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

	+	000 0	001 1	010 x	011 $x + 1$
00	0	0	1	x	$x + 1$
00	1	1	0	$x + 1$	x
01	x	x	$x + 1$	0	1
01	$x + 1$	$x + 1$	x	1	0

	×	000 0	001 1	010 x	011 $x + 1$
00	0	0	0	0	0
00	1	0	1	x	$x + 1$
01	x	0	x	$x + 1$	1
01	$x + 1$	0	$x + 1$	1	x

5.11 $x^2 + 1$

5.12 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 6 ADVANCED ENCRYPTION STANDARD

ANSWERS TO QUESTIONS

6.1 **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.

6.2 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.

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

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

6.5 **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 *i*th 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.

6.6 For SubBytes, 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.

6.7 For ShiftRows, 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.

6.8 12 bytes.

6.9 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.

6.10 For AddRoundKey, 128 bits of **State** are bitwise XORed with the 128 bits of the round key.

6.11 The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). The expansion is defined by the pseudocode in Section 6.4.

6.12 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.

6.13 ShiftRows is described in the answer to Question 6.8. 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.

6.14 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

6.1 We want to show that d(x) = a(x) x b(x) mod (x^4 + 1) = 1. Substituting into Equation (6.14) in Appendix 6A, 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

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

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

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

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

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

6.2 a. {01}

b. We need to show that the transformation defined by Equation 6.2, 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 6.2a).

6.3 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}

6.4

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	B2
55	E6	04	22
3E	2E	B8	8C
10	15	58	0A

e

6.5 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.

6.6 **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

6.7 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.

6.8

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

6.9 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 1 + 1 + 1 + 2 =5.

6.10 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

6.11 $$\begin{bmatrix} x^3+1 & x \\ x & x^3+1 \end{bmatrix} \begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

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

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

6.13 For convenience, we drop the "j" subscript. We show the equivalence for the first equation; the rest are shown in the same fashion. From Equation (6.8), we have

$$s'_0 = (2 \bullet s_0) \oplus (3 \bullet s_1) \oplus s_2 \oplus s_3$$

From Equation (6.9), we have

s'_0	$= s_0 \oplus \text{Tmp} \oplus [2 \bullet (s_0 \oplus s_1)]$	
	$= s_0 \oplus (s_0 \oplus s_1 \oplus s_2 \oplus s_3) \oplus [2 \bullet (s_0 \oplus s_1)]$	substituting for Tmp
	$= s_0 \oplus s_0 \oplus s_1 \oplus s_2 \oplus s_3 \oplus (2 \bullet s_0) \oplus (2 \bullet s_1)$	expanding the final term
	$= s_0 \oplus s_0 \oplus (2 \bullet s_0) \oplus s_1 \oplus (2 \bullet s_1) \oplus s_2 \oplus s_3$	rearranging terms
	$= (2 \bullet s_0) \oplus s_1 \oplus (2 \bullet s_1) \oplus s_2 \oplus s_3$	cancelling first two terms
	$= (2 \bullet s_0) \oplus (3 \bullet s_1) \oplus s_2 \oplus s_3$	using the identity referenced just before Equation (6.9)

Chapter 7 Block Cipher Operation

Answers to Questions

7.1 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.

7.2 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.

7.3 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.

7.4 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.

7.5 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

7.1 **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, X_1 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.

7.2 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.

7.3 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.

7.4 **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.

7.5 In CBC encryption, the input block to each forward cipher operation (except the first) depends on the result of the previous forward cipher operation, so the forward cipher operations cannot be performed in parallel. In CBC decryption, however, the input blocks for the inverse cipher function (i.e., the ciphertext blocks) are immediately available, so that multiple inverse cipher operations can be performed in parallel.

7.6 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.

7.7 For this padding method, the padding bits can be removed unambiguously, provided the receiver can determine that the message is indeed padded. One way to ensure that the receiver does not mistakenly remove bits from an unpadded message is to require the sender to pad every message, including messages in which the final block is already complete. For such messages, an entire block of padding is appended.

7.8 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.

7.9 Let message M1 have plaintext blocks $P1_j$ and ciphertext blocks $C1_j$. Similarly for message M2. If the same IV and key are used in OFB mode for both messages, then both messages have the same output blocks O_j.Suppose an attacker can observe the ciphertext blocks for M1 and M2 and that the attacker knows the exact contents of $P1_q$.
Then,

$C1_q = P1_q \oplus O_q$	by definition of OFB
$C1_q \oplus P1_q = P1_q \oplus O_q \oplus P1_q$	add to both sides
$O_q \oplus P1_q \oplus P1_q = C1_q \oplus P1_q$	rearrange
$O_q = C1_q \oplus P1_q$	cancel terms
$C2_q = P2_q \oplus O_q$	by definition of OFB
$C2_q \oplus O_q = P2_q \oplus O_q \oplus O_q$	add to both sides
$P2_q = C2_q \oplus O_q$	add to both sides

7.10 $O_i = C_i \oplus P_i$

7.11 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}.

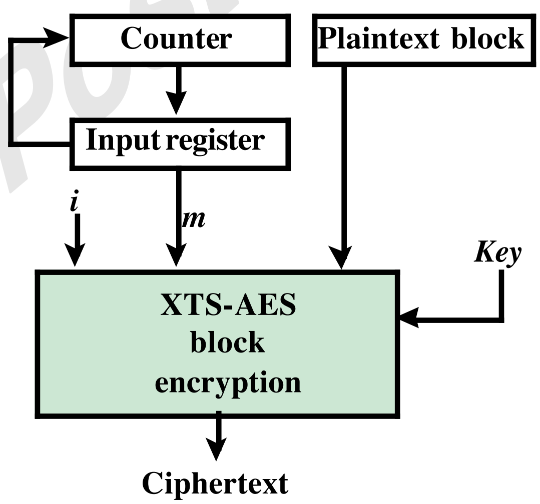
The last two blocks of the ciphertext are C_{N-1} and C_N.

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

7.12 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.

7.13



7.14 a. (i) 13 (ii) 14 (iii) 16 (iv) 17 (v) 18 (vi) 18 (vii) 14 (viii) 14
Explanation: For problem (i), $2^{12} < 8191_{10} < 2^{13}$. Hence 8191_{10} can be accommodated in a 13-bit integer but not in a 12-bit integer. For problem (ii), $2^{13} \leq 8192_{10} < 2^{14}$, i.e. 8192_{10} cannot be accommodated into a 13-bit integer, but can be represented in 14

bits. Similar arguments apply to problems (iii), (iv), and (viii). In problems (v), (vi), and (vii), the numbers are represented using hexadecimal digits. Convert to decimal and use the above method to determine the number of bits.

b. (i) 2 (ii) 2 (iii) 2 (iv) 3 (v) 3 (vi) 3 (vii) 2 (viii) 2

Explanation: For each problem, $B = \left\lceil \frac{b}{8} \right\rceil$ where *b* is the bit-length and *B*, the resulting byte-length of a binary number. Using this equation to compute *B* from *b* representing each of the answers (i)-(viii) of Problem (2a) leads to the above answers.

7.15 **a.** 10 4 23 5; 17 0 13 3 14 12 10; 14 17 4; 3 8 6 8 19 6
b. 6887; 202214284; 381; 1515819
c. 1A E7; 0C 0D 8B 8C; 01 7D; 17 21 2B
d. 3; 6; 2; 5
e. It is equal to the length of the character string.

7.16 **a.** **(i)** If n is even then x = n/2; If n is odd then x is the largest integer less than n/2, which is (n − 1)/2
(ii) x is an integer, so y = n − x. The result follows.
(iii) If x = n/2, then y = n/2 and y = x
If x = (n − 1)/2, then y = (n + 1)/2 and y > x
b. If n is even, LEN(A) = LEN(B); otherwise LEN(A) = LEN(B) − 1

7.17 b and d are byte lengths. When B is encoded as a byte string in Step 5i, b is the number of bytes in the encoding. The definition of d ensures that the output of the Feistel round function is at least four bytes longer than this encoding of B, which minimizes any bias in the modular reduction in Step 5vi.

7.18 SP800-38G states that The length of the key affects its resistance to brute-force search. The requirement for each mode that $radix^{minlen} \geq 100$ precludes a generic meet-in-the-middle attack on the Feistel structure. A requirement on maxlen for FF2—namely, that maxlen ≤ $2\lfloor 98/\mathrm{LOG}_2(radix)\rfloor$ if radix is not a power of 2—minimizes the bias in the generation of z.
Otherwise, the choices of the mode parameters, e.g., radix, minlen, and maxlen, are determined by the needs of the application, not by security considerations. For FF2, with a maximum radix value of 2^8, the radix value can be stored in one byte. FF2 also supports the shortest plaintext length and that may have influenced the decision on radix value.

Chapter 8 Random and Pseudorandom Number Generation and Stream Ciphers

Answers to Questions

8.1 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.

8.2 **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.

8.3 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.

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

Answers to Problems

8.1 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

8.2 **a.** Maximum period is $2^{4-2} = 4$
b. *a* must be 3, 5, 11, or 13
c. The seed must be odd

8.3 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.

8.4 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.

8.5 Many packages make use of a linear congruential generator with $m = 2^k$. As discussed in the answer to the preceding problem, 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.

8.6 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.

8.7 **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.

8.8 **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 RC4($v \,||\, k$) ⊕ 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.

8.9 The pattern repeats after 15 bits. Reading the bits from right to left, it matches the bit pattern in Figure 8.9b.

$$1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0$$

$$1 + X + X^2 + X^3 + X^5 + X^7 + X^8 + X^{11} + X^{15} + \cdots$$

$$1 + X + X^3 \;\big)\; 1$$

$$
\begin{array}{l}
\underline{1 + X + X^4} \\
X + X^4 \\
\underline{X + X^2 + X^5} \\
X^2 + X^4 + X^5 \\
\underline{X^2 + X^3 + X^6} \\
X^3 + X^4 + X^5 + X^6 \\
\underline{X^3 + X^4 + X^7} \\
X^5 + X^6 + X^7 \\
\underline{X^5 + X^6 + X^9} \\
X^7 + X^9 \\
\underline{X^7 + X^8 + X^{11}} \\
X^8 + X^9 + X^{11} \\
\underline{X^8 + X^9 + X^{12}} \\
X^{11} + X^{12} \\
\underline{X^{11} + X^{12} + X^{15}} \\
X^{15}
\end{array}
$$

8.10 a

Feedback bit	State of shift register					Output bit
	1	0	0	0	0	
0	0	1	0	0	0	0
1	1	0	1	0	0	0
0	0	1	0	1	0	0
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	0	1	0
0	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1
0	0	1	1	0	1	1
0	0	0	1	1	0	1
0	0	0	0	1	1	0
1	1	0	0	0	1	1
1	1	1	0	0	0	1
1	1	1	1	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
0	0	1	1	1	1	1
0	0	0	1	1	1	1
1	1	0	0	1	1	1
1	1	1	0	0	1	1
0	0	1	1	0	0	1
1	1	0	1	1	0	0
0	0	1	0	1	1	0
0	0	0	1	0	1	1
1	1	0	0	1	0	1
0	0	1	0	0	1	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0
0	0	0	0	0	1	0
1	1	0	0	0	0	1

8.10 b

Feedback bit	State of shift register					Output bit
	1	0	0	0	0	
1	1	1	0	0	0	0
0	0	1	1	0	0	0
1	1	0	1	1	0	0
0	0	1	0	1	1	0
1	1	0	1	0	1	1
0	0	1	0	1	0	1
0	0	0	1	0	1	0
1	1	0	0	1	0	1
0	0	1	0	0	1	0
0	0	0	1	0	0	1
0	0	0	0	1	0	0
1	1	0	0	0	1	0
0	0	1	0	0	0	1
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0
1	1	1	1	1	0	1
1	1	1	1	1	1	0
0	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
0	0	1	1	0	1	1
0	0	0	1	1	0	1
1	1	0	0	1	1	0
1	1	1	0	0	1	1
1	1	1	1	0	0	1
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	0	1	1	1
0	0	0	0	0	1	1
1	1	0	0	0	0	1

8.11

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.

8.12 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.

8.13 Sixty-five thousand.

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:

$$C = \mathrm{E}(PU_b, M)$$

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

$$M = \mathrm{D}(PR_b, \mathrm{C}) = \mathrm{D}(PR_b, \mathrm{E}(PU_b, M))$$

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

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 = 14.
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 modulo $\phi(n)$ 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 ☐(N) (where '☐ 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 mod 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 ≠ 1 and 222 ≠ 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 ≠ 1
- 167 ≠ 237 – 1
- 167^2 mod 237 = 160 ≠ 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 ≠ 1 and 16 ≠ 241 – 1

$16^2 \bmod 241 = 256 \bmod 241 = 15$
$15 \neq 241 - 1$
$15^2 \bmod 241 = 225 \bmod 241 = 225$
$225 \neq 241 - 1$
$225^2 \bmod 241 = 15$
$15 \neq 241 - 1$
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:
$d_p = d \bmod (p - 1)$ $d_q = d \bmod (q-1)$
$d_p = 19367 \bmod 232 = 111$ $d_q = 19367 \bmod 240 = 167$

$C_p = C \bmod p$
$M_p = C_p^{d_p} \bmod p = 141^{111} \bmod 233 = 2$
$C_q = C \bmod q$
$M_q = C_q^{d_q} \bmod q$
$M_q = 121^{167} \bmod 241 = 2$
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 $d = e^{-1} \bmod \phi(n)$. M2 consists of raising M to the power e (mod n). M3 consists of raising C to the power d (mod n).

9.13 Yes.

9.14 This algorithm is known as Cocks algorithm.

a. Cocks makes use of the Chinese remainder theorem (see Section 8.4 and Problem 8.17), 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 $de \equiv 1 \bmod ((P - 1)(Q - 1))$, 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 9.8. 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:

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

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

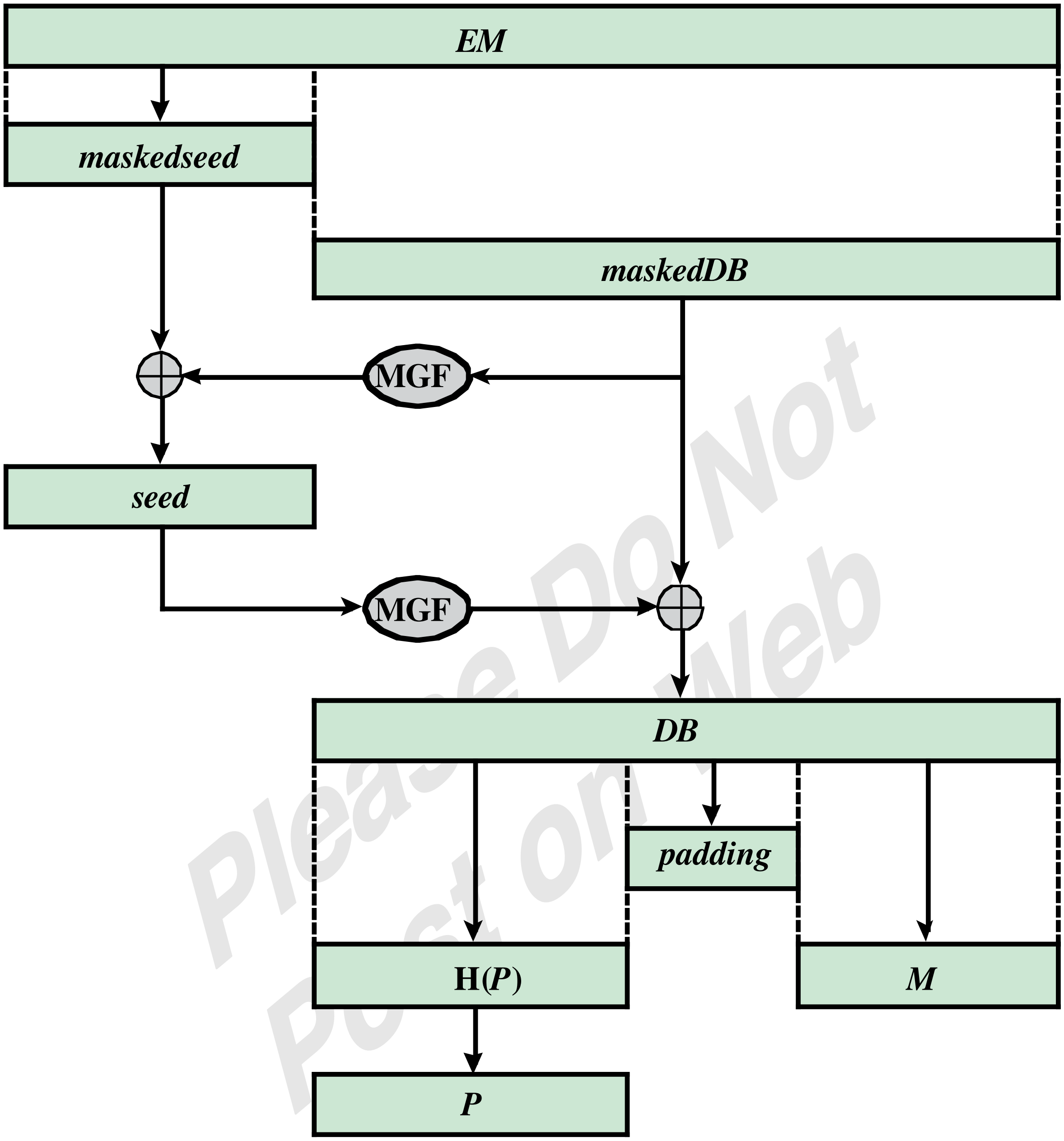
The algorithm preserves the invariant that $d = a^c \bmod n$ as it increases c by doublings and incrementations until $c = b$.

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



CHAPTER 10 OTHER PUBLIC-KEY CRYPTOSYSTEMS

ANSWERS TO QUESTIONS

10.1 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.

10.2 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

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

10.4 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 $x_A^g x_B^g = \left(x_A x_B\right)^g$. 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 $K2 = \left(Y_A\right)^{X_D} \bmod q$
4. Bob receives Y_D and calculates $K1 = \left(Y_D\right)^{X_B} \bmod q$.
5. Bob transmits X_A to Alice.
6. Darth intercepts X_A and transmits Y_D to Alice. Darth calculates $K1 = \left(Y_B\right)^{X_D} \bmod q$.
7. Alice receives Y_D and calculates $K2 = \left(Y_D\right)^{X_A} \bmod q$.

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

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

10.8 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 $4(1)^3 + 27(1) = 31 \neq 0$.

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

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

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

$\Delta = (8 - 9)/(-2 + 3) = -1$
$x_R = 1 + 3 + 2 = 6$
$y_R = -9 - (-3 - 6) = 0$
$R = (7, 2)$

b. For $R = 2P$, we use Equations (10.4), with $a = -36$
$x_r = [(27 - 36)/18]^2 + 6 = 25/4$
$y_R = [(27 - 36)/18](-3 - 25/4) - 9 = 35/8$

10.11 $(4a^3 + 27b^2) \bmod p = 4(10)^3 + 27(5)^2 \bmod 17 = 4675 \bmod 17 = 0$
This elliptic curve does not satisfy the condition of Equation (10.6) and therefore does not define a group over Z_{17}.

10.12

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

10.13 The negative of a point $P = (x_P, y_P)$ is the point $-P = (x_P, -y_P \bmod p)$. Thus

$-P = (5,9)$; $-Q = (3,0)$; $-R = (0,11)$

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

$$\lambda = (3 \times 2^2 + 1)/(2 \times 7) \bmod 11 = 13/14 \bmod 11 = 2/3 \bmod 11 = 8$$

Then we have

$x_3 = 8^2 - 2 - 2 \bmod 11 = 5$
$y_3 = 8(2 - 5) - 7 \bmod 11 = 2$
$2G = (5, 2)$

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)

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

10.16 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.

10.17 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.