SOLUTIONS TO PRACTICE PROBLEMS

CRYPTOGRAPHY AND NETWORK SECURITY

PRINCIPLES AND PRACTICE

SIXTH EDITION

WILLIAM STALLINGS

TABLE OF CONTENTS

Chapter 1 Introduction 3
Chapter 2 Classical Encryption Techniques..... 4
Chapter 3 Block Ciphers and the Data Encryption Standard..... 6
Chapter 4 Basic Concepts in Number Theory and Finite Fields 8
Chapter 5 Advanced Encryption Standard..... 10
Chapter 6 Block Cipher Operation..... 11
Chapter 7 Pseudorandom Number Generation and Stream Ciphers12
Chapter 8 Introduction to Number Theory 14
Chapter 9 Public-Key Cryptography and RSA 15
Chapter 10 Other Public-Key Cryptosystems 17
Chapter 11 Cryptographic Hash Functions 18
Chapter 12 Message Authentication Codes..... 19
Chapter 13 Digital Signatures 20
Chapter 14 Key Management and Distribution..... 21
Chapter 15 User Authentication 22
Chapter 17 Transport-Level Security..... 24
Chapter 18 Wireless Network Security 25
Chapter 19 Electronic Mail Security..... 26
Chapter 20 IP Security..... 27
Chapter 21 Malicious Software 28
Chapter 22 Intruders 29
Chapter 23 Firewalls..... 30

CHAPTER 1 INTRODUCTION

1.1 All of these activities could create the right conditions to threaten the network.

a. The regular daily courier is familiar to employees, so they may not notice anything is wrong should that person walk into the server room.

b. Even with good severance packages and benefits, employees who lost their jobs due to downsizing may be disgruntled.

c. An employee's traveling to another location may not create a threat, but if the employee has a laptop computer that contains private information or the Web browser has saved passwords, then if the laptop is stolen, a hacker has gained valuable information.

d. If the sprinkler system went off, it could damage the company's servers and other computing equipment.

CHAPTER 2 CLASSICAL ENCRYPTION TECHNIQUES

2.1 If we shift by a multiple of the key length, the probability of coincidence is

$$(.6)2 + (.3)2 + (.1)2 = 0.46.$$

So we would expect about $1000 \times 0.46 = 460$ coincidences. Other shifts would give lower indices of coincidence.

2.2 The one time pad system requires that we secretly communicate in advance a key which is at least as long as the message we will send. This is a severe practical difficulty since it requires substantial secret communication in advance of the desired secret communication.

2.3 An affine cipher has the form $y \equiv ax+b$ where x is the plaintext and y is the ciphertext (both integers modulo 26). We need to find a and b. Converting to numbers, the plaintext is 14, 14, 15, 18 and the ciphertext is 1, 1, 3, 9. Thus we need to solve the equations $14a + b = 1$ and $15a + b = 3$. Subtracting the equations, we find $a = 2$ and plugging this into either equation gives $b = 25$.

2.4 Observe that A = 0 gets mapped to H = 7, so $7 = \beta$. Now, B = 1 gets mapped to O = 14 = $\alpha + \beta$ so $\alpha = 7$.

2.5 To estimate the period we use the Kasiski test. The distance between the two occurrences given is $241 - 10 = 231 = 3 \times 7 \times 11$ positions. Possible periods are thus 3, 7 and 11. If the guess is correct, we can immediately find the corresponding shifts: at position 10 the shift is T – c = 19 – 2 = 17 = r. Similar computations for the other positions give the shift keys *rrectcorrect*. We now see that this is not periodic with periods 3 or 11, while period 7 is possible. The keyword of length 7 starts at position 15; hence the keyword is *correct*.

2.6 This does not add much security to the system at all. Capital letters usually appear only at the beginning of words at the beginning of sentences. Thus, the frequencies of capital letters are quite small in English text. You could simply consider this while using frequency analysis. Simply put, disregard all the characters of very small frequencies and concentrate on solving for the characters with the highest frequencies, which will still be the same lowercase letters. Once

these are solved for, there will be enough recovered plaintext to deduce most if not all of the capital letters in the message.

2.7 The described situation is
$C_1 = K \oplus M_1$
$C_2 = K \oplus M_2$.
From this we get
$C_1 \oplus C_2 = (K \oplus M_1) \oplus (K \oplus M_1) = M_1 \oplus M_2$
i.e. the Adversary can identify exactly in which bit positions the two messages differ.

2.8 Let the encryption matrix be $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then we have the following equations:
$12a + 5b = 14 \bmod 17$
$12c + 5d = 10 \bmod 17$

Solve these equations for *a* and *c* respectively:

$12a = (14 - 5b) \bmod 17$
$a = (12^{-1} \bmod 17)(14 - 5b) \bmod 17$
$12c = (10 - 5d) \bmod 17$
$c = (12^{-1} \bmod 17)(10 - 5d) \bmod 17$

Now, notice that plugging in each of the 17 possible values of *b* yields a solution for *a* and plugging in each of the 17 possible values of *d* yields a solution for *c*. Thus, the total possible number of keys seems to be 17 × 17 = 289. But, some of these keys give a determinant equivalent to 0 mod 17, which is not permissible. To see this, using the equations above and solving for the determinant *ad–bc* eventually yields the expression: $8 + 2b + 4c$. We can set this to 0, and quickly show that there are 17 combinations of values for b and c that have this equal to 0. (Once *b* and *c* are set, so are *a* and *d*.) Thus, we must subtract 17 from our original answer to yield a final answer of 272 possible total keys. To find two of these, simply plug in two sets of values for a and b above and one set of values for c and d. For example:
$a = (4 + b) \bmod 17$, two solutions to this are $a = 5$, $b = 1$ and $a = 6$, $b = 2$.
The other equation simplifies to $d = (2 + c) \bmod 17$. One solution to this is
d = 3, c = 1. So two possible keys are $\begin{pmatrix} 5 & 1 \\ 1 & 3 \end{pmatrix}$ and $\begin{pmatrix} 6 & 2 \\ 1 & 3 \end{pmatrix}$

CHAPTER 3 BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

3.1 Kirchhoff's principle says that one should always assume that the attacker knows the algorithm being used. A designer who believes this principle will circulate his or her algorithm widely and it will be tested by many talented cryptanalysts. A designer who does not believe the principle may try to keep the algorithm secret. Thus the algorithm will only be tested by a few cryptanalysts and we can be much less confident of its strength.

3.2 The key space has 2^{40} elements, so brute force would take 2^{20} seconds, which is about 12 days. This would be practical if the message revealed the location of enemy missiles in a cold-war situation. It would be impractical if the message's useful life was very short, for example if it was a few frames in a pay-per-view sports video. Doubling the key size would make the brute force decryption time 2^{60} seconds, which is about 3.8×10^{16} years. There is no scenario in which this would be practical.

3.3 The left 28 bits of the key affect only the S-boxes S_1, ... , S_4. In the modified DES above, changing these key values affects only the left 16 bits of the output of the internal f-function. This is due to the fact there is no permutation following the S-boxes, and the expansion function also keeps all the bits in the "right" place. That is, the 48 bits coming out of the expansion function are such that each half is derived from its respective half of the input. The left 24 bits that are derived from the left 16 bits of the input affect only the S-boxes S1, …, S4, and so once again all changes are local to the left 16 bits coming out of the DES internal f-function. (In short, the modification has completely ruined the DES avalanche effect.) It is therefore possible to exhaustively search over each half of the key separately. The complexity of the attack is approximately 2^{28} for each half of the key; all in all, it requires approximately 2^{30} local DES computations.

3.4 **a.** The equality states that the uncertainty of the ciphertext is zero when plaintext and key are given. Since, in the random cipher model, the ciphertext is a function of plaintext and key, this holds.

b. We have the following sequence of (in)equalities:

$$\begin{aligned} H(C, M, K) &= H(C|M, K) + H(M, K) \\ &= 0 + H(M, K) \end{aligned}$$

$\leq (H(M) + H(K))$

where we have used first the general equality $H(A, B) = H(A|B) + H(B)$, then the equality from (a) and finally the general inequality $H(A, B) \leq H(A) + H(B)$

c. Equality holds in the last inequality iff M and K are independent. It seems natural to expect from a ciphersystem that key is chosen independently from the plaintext and thus to have equality in (b)

3.5 a. Let $c = DESX(K, m)$. We first xor both sides with k_2, which gives $c \oplus k_2 = DES(k_1, (m \oplus k_2))$. Next, we apply DES decryption, to get $DES^{-1}(k_1, (c \oplus k_2)) = DES^{-1}(k_1, [DES(k_1, (m \oplus k_2)]) = m \oplus k_2$. Finally, we again XOR both sides with $k_2$to get the final result

$$m = DES^{-1}(k_1, (c \oplus k_2)) \oplus k_2$$

b. DESX' can be attacked using a meet-in-the-middle attack. We assume that the adversary has a few plaintext/ciphertext pairs (m, c). He can then do a brute force attack on the DES part, i.e. compute $x = DES(k_1, m)$ for all possible keys k_1, and store the resulting pairs (x, k_1) in a dictionary. Then he goes through all possible $k_2$values, computes $c \oplus k_2$ and looks it up in the dictionary. When found, he has a potential key pair (k_1, k_2). The complexity of this attack is 2^{64}, which shows that the cipher does not provide 120 bits of security.

Alternatively, with two plaintext/ciphertext pairs (m_1, c_1) and (m_2, c_2), one notes that $c_1 \oplus c_2 = DES(k_1, m_1) \oplus DES(k_1, m_2)$, which makes it possible to do a brute force attack with only twice the cost of an attack against DES.

CHAPTER 4 BASIC CONCEPTS IN NUMBER THEORY AND FINITE FIELDS

4.1 We have
$654 = 5 \times 123 + 39$
$123 = 3 \times 39 + 6$
$39 = 6 \times 6 + 3$
$6 = 2 \cdot 3 + 0$
and so the gcd $d = 3$. Working backwards, we have
$3 = 39 - 6 \times 6$
$= 39 - 6(123 - 3 \times 39) = 19 \times 39 - 6 \times 123$
$= 19(654 - 5 \times 123) - 6 \times 123 = 19 \times 654 - 101 \times 123$
and so $m = 19$ and $n = -101$.

4.2 $123 \equiv 18 \pmod{35}$ so $123^{241} \equiv 18^{241} \pmod{35}$. Also note that $\phi(35) = (5 - 1)(7 - 1) = 24$. Thus, for a given a relatively prime with 35, we have $a^{24} \equiv 1 \pmod{35}$. Thus,
$18^{241} = 18(18^{24})^{10} \equiv 18(1)^{10} \equiv 18 \bmod 35$.

4.3 Suppose $a \neq 0$. If $d|a$, the $|d| \leq |a|$, so there are at most $2|a| + 1$ choices for d.

4.4 **1.** First we divide $a(x)$ by $b(x)$ to get a remainder $r_1(x) = x^8 + x^7 + x^6 + x^2 + x$
2. Then we divide $b(x)$ by $r_1(x)$ to get a remainder $r_2(x) = x^5 + x^2 + x + 1$
3. Then we divide $r_1(x)$ by $r_2(x)$ to get a remainder $r_3(x) = x^3 + x + 1$
4. Then we divide $r_2(x)$ by $r_3(x)$ to get a remainder $r_4(x) = 0$
Therefore, $\gcd(a(x), b(x)) = r_3(x) = x^3 + x + 1$

4.5 **a.** $x = 4$
b. no solution
c. no solution
d. no solution

4.6 **a.** 2
b. 7
c. nonexistent
d. 8

e. 5
f. nonexistent
g. 25
h. 7
i. 5982

CHAPTER 5 ADVANCED ENCRYPTION STANDARD

5.1 The state in AES is a 4 × 4 matrix with entries in the field of 256 elements. The shift row layer shifts each row to the right a certain amount, wrapping the entries around. More precisely, the first row is not shifted, the second row is shifted by one, the third row is shifted by two, and the fourth row is shifted by three.
The Byte Substitution layer can be viewed as a lookup table. Each matrix entry, represented by an 8-bit byte, is broken into two pieces which index the rows and columns of a 16 × 16 lookup matrix. The byte is replaced by the corresponding entry in the table, which is another 8-bit byte. The Byte Substitution layer is applied entry by entry to the state, with all entries treated in the same way. The Row Shift layer simply moves the bytes around. Thus it doesn't matter in which order we apply these layers: shifting and substituting is the same as first substituting then shifting.

5.2 w(0) = {11 11 11 11}; w(1) = {11 11 11 11}; w(2) = {11 11 11 11}; w(3) = {11 11 11 11};

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i – 4]	w[i] = temp ⊕ w[i – 4]
4	11111111	11111111	16161616	01000000	17161616	11111111	E8E9E9E9
5	E8E9E9E9					11111111	17161616
6	17161616					11111111	E8E9E9E9
7	E8E9E9E9					11111111	17161616

CHAPTER 6 BLOCK CIPHER OPERATION

6.1 there are 26 possibilities for *b* and 12 possibilities for *a* (see Problem 2.1 in book). Let $E_2(a, x) = ax \bmod 26$ and let $E_1(b, x) = x + b \bmod 26$. The composition of these two gives the affine cipher. The total computation needed involves producing 26 encryptions for E_2 and 12 decryptions for E_1. The total comes out to be 38.

6.2 DES takes a 8-octet (64-bit) plaintext block and yields a 8-octet cipher block. CBC requires a 8-octet initialization vector (IV) to be sent along with the cipher blocks. So X now sends 64 octets of cipher blocks plus 8 octets of IV, for a total of 72 octets.

CHAPTER 7 PSEUDORANDOM NUMBER GENERATION AND STREAM CIPHERS

7.1 a. An n-bit LFSR has 2^n possible states. The output depends only on the state, so whenever the device returns to a previous state, we will have completed one period. Thus the period is at most 2^n. However, the all-zero state cannot appear in a maximal period sequence, since it would generate an all zero output.

b. If an LFSR has an odd number of taps and enters the state with all ones, then the bit to be shifted in is the XOR of an odd number of ones and hence one. So, the device is stuck in this state.

7.2 Inspection of the output sequence shows that it seems to have period 21; at least that holds for the given part. Thus an LFSR that produces this sequence must have length at least 5: a LFSR of length 4 can produce a period of at most $2^4 - 1 = 15$; an LFSR of length 5 can produce a period of up to length $2^5 - 1 = 31$. Trying with an LFSR of this length with tap sequence $c_1c_2c_3c_4c_5$, we can make the following deductions. The first five output bits are 00100, therefore the register is initialized with 00100. The sixth output bit is 0 and is a function of the initial set of register bits; therefore c_3 must be 0, otherwise, a 1 bit would be fed into the left side of the register and be output as the sixth bit. After the first bit is output, the register contents are 00010. Because the seventh bit of output is 0, by similar reasoning, we must have $c_4 = 0$. The eighth output bit is 1 and by the similar reasoning, c_5 must be 1. By this point, the contents of the register is 10000 and the tenth output bit is 1, so we must have $c_1 = 1$. By this point, the contents of the register is 11000 and the eleventh output bit is 1, so we must have $c_2 = 0$ because $c_1 = 1$ and we must have $c_1 \oplus c_2 = 1$. It remains to check that this LFSR actually generates the given periodic output with period 21. Computing the first 21 outputs verifies this.

7.3 a. Message length is $8 \times 25 = 200$ bits. The period length for an LFSR with size L bits is at most $2^L - 1$. To achieve a period of at least 200 bits we must thus choose $L > 8$. The LFSR size is also the size of the key (the initial content), so minimum key size is 8.

b. The adversary can construct the first 40 bits of the message m and thus determine the first 40 bits of the keystream (as $b = c \oplus m$). But

to completely determine the tap sequence of an 8 bit LFSR it is enough to know $2 \times 8 = 16$ consecutive bits of output. So the adversary can decrypt Alice's messages.

7.4 Period is 7, so the length must be at least 3. Let the tap sequence be $c_1c_2c_3$. From the given output sequence we can form the system of equations

$$0c_1 \oplus 0c_2 \oplus 1c_3 = 1$$
$$1c_1 \oplus 0c_2 \oplus 0c_3 = 1$$
$$1c_1 \oplus 1c_2 \oplus 0c_3 = 1:$$

This is directly solved from top to bottom, giving $c_3 = 1$, $c_1 = 1$, $c_2 = 0$.

We also need to check that this LFSR actually produces the given output, i.e. that the seventh bit of output is 0. Since we tap at positions 1 and 3, the seventh bit is $1 \oplus 1 = 0$.

7.5 In a maximal period LFSR (period length $2^n - 1$), all non-zero states are traversed in a period. The output bit in each state is the first bit in the state. Thus there are
2^{n-1} ones and $2^{n-1} - 1$ zeros in a period; the omitted all-zero state corresponds to the subtraction of 1 from the number of zeros.

7.6 1

CHAPTER 8 INTRODUCTION TO NUMBER THEORY

8.1 The answer is no. Proof: Consider any three consecutive odd numbers, together with the two even numbers between the odd numbers. In any such sequence of five consecutive integers, beginning with an odd integer, one of the odd numbers has to be divisible by three.

8.2 **a.** We want that $(m^e)^d = m^{ed} = m \in Z_p$. For this we need $ed = 1$ in Z_{p-1}. So, we compute $d = e^{-1}$ in Z_{p-1} using the extended Euclidean algorithm, which is possible if $\gcd(e, p - 1) = 1$.

b. The difference is that in Pohlig-Hellman $\Phi(p)$ is known for everyone, so anyone who knows e can compute d. In RSA, $\Phi(N) = (p-1)(q-1)$, so you must know the factorization of N to compute d from e.

8.3 If all p|n are greater than $\sqrt{n}$, then n is a product of at least two primes p_1 and p_2. Therefore $n \geq p_1 p_2 > \sqrt{n}\sqrt{n} = n$, which is a contradiction.

8.4 If and only if n is a multiple of 3.

8.5 17

8.6 Calculate $a^6 \bmod 7$ for $a = 1, 2, 3, 4, 5, 6$. You should get 1 for these values of a.

8.7 If we have an integer n and a is less than n with $a^{n-1} \bmod n \neq 1$, then we know n is not prime. So, for a given large value of n, if we calculate $a^{n-1} \bmod n$ for many values of $a < n$ and always get 1, we suspect that n is prime.

8.8 We need to find a value of a such that $a^9 \bmod 10 \neq 1$. Any integer a where $1 < a < 10$ will work.

CHAPTER 9 PUBLIC-KEY CRYPTOGRAPHY AND RSA

9.1 **a.** B would simply do $y^{k_3} \bmod n = m^{k_1 k_2 k_3} \bmod n$. By Euler's theorem, $m^{k_1 k_2 k_3} \bmod n = m \bmod n$. C would do the same.

b. Observe that A and B can encrypt their message successively. First A computes $y = m^{k_1 k_2} \bmod n$. Then B computes $z = y^{k_2 k_3} \bmod n = \left(m^{k_1 k_2}\right)^{k_2 k_3}$

Now, if A and B loses m, then they can't recover m since neither of them, by themselves, knows both k1 and k3.???

9.2 It is required that gcd (e, $\phi(n)$) = 1 for the RSA system to work properly. But $\phi(n) = (p - 1)(q - 1)$ is an even number, so gcd(2, $\phi(n)$) = 2. In particular, it would not be possible to choose d with $ed \equiv 1 \pmod{\phi(n)}$.

9.3 The general argument against double encryption is that it is subject to the meet-in-the middle attack, which has time complexity similar to that of a single brute force attack. In the particular case of RSA encryption, double encryption is also meaningless, since the double encryption is equivalent to a single RSA encryption with public key $e1 \times e2$ and private key $d1 \times d2$. To verify this, see that

$$(m^{e1} \bmod n)^{e2} \bmod n = m^{e1e2} \bmod n.$$

9.4 Counter mode is unusable, since here decryption is the same as encryption and anyone can encrypt messages, using the receiver's public key. Hence anyone intercepting a ciphertext can decrypt it. CBC mode does not have this disadvantage, since it uses the decryption function.

9.5 We know that $e_1 d_1 \equiv 1 \bmod \backslash(n)$ and $e_2 d_2 \equiv 1 \bmod \backslash(n)$. Thus, we also know that $\backslash(n) \mid (e_1 d_1 - 1)$ and $\backslash(n) \mid (e_2 d_2 - 1)$. Since we know e_1, d_1, e_2, and d_2. We can calculate the quantities $(e_1 d_1 - 1)$ and $(e_2 d_2 - 1)$. We also know that
$\gcd((e_1 d_1 - 1), (e_2 d_2 - 1)) \geq \backslash(n)$. Thus, we can simply run Euclid's algorithm on the two numbers, and get an output value that is some multiple of $\backslash(n)$. With any luck, it will be a small multiple and we can guess the value of $\backslash(n)$, allowing us to determine d_3, from e_3.

9.6 a. We want that $(M^e)^d \bmod p = M^{ed} \bmod p = M$. For this we need $ed \equiv 1 \pmod{(p-1)}$. So, we compute $d = e^{-1}$ using the extended Euclidean algorithm, which is possible if gcd(e, $p - 1$) = 1. Because, p is prime, this is true for all e less than p.

b. The difference is that in Pohlig-Hellman $\phi(p)$ is known for everyone, so anyone who knows e can compute d. In RSA, $\phi(n) = (p-1)(q-1)$, so you must know the factorization of n to compute d from e.

9.7 No. We require that $ed = 1 \bmod \phi(n)$, where d is the decryption exponent. But
$\phi(n) = (p-1)(q-1)$ is an even number, so if e is even, we cannot find such a d.

9.8 The adversary has eavesdropped and thus knows $C = M^e$ and $C' = M^{e'}$. He also knows e and e'. Furthermore, gcd (e, e') = 1, because $e' = e + 2^i$ for some i. (Any non-trivial divisor of e must be odd, hence not a divisor of 2^i, hence not a divisor of e'.) So the adversary can find integers x and y such that $ex + e'y = 1$. Hence
$C^x \times C'^y = M^{ex+e'y} = M$.

9.9 a. Mallory's assumption is that Alice's message is $10x$ for some integer x. Then we have $c = (10m)^e = 10^e m^e$, where the computations are modulo n. Mallory can compute 10^e and invert it using the extended Euclidean algorithm to get 10^{-e}. Finally, he constructs the bid $c \times (10)^{-e} \times 11^e$, which equals $(11m)^e$, i.e. the encryption of $11m$.

b. Two main ingredients in padding are randomization (to avoid that the same message encrypted twice gives the same encryption) and redundancy (so that randomly constructed cipher texts are unlikely to be encryptions of a valid message)

9.10 $n = (p-1)(q-1) = 10{,}200 = 2^3\ 3\ 5^2\ 17$. e can be any integer relatively prime to n, such as 7, 143, 689.

CHAPTER 10 OTHER PUBLIC-KEY CRYPTOSYSTEMS

10.1 **a.** To decrypt, the receiver first parses the ciphertext as $Y \| C \| T$. She then computes $Y^x = K$, and then $k_1 \| k_2 = \mathrm{H}(K \| Y)$. She can now check whether
$T = \mathrm{MAC}(k_2, C)$; if not the ciphertext is rejected. If the MAC is valid, the final step is to decrypt the message as $m = \mathrm{D}(k_1, C)$.

b. A cipher is CCA2 resistant (resistant against adaptive chosen ciphertext attacks) if an efficient adversary has no more than a negligible advantage over guessing in the following game against Alice:
•The adversary chooses two ciphertext m and m` and gives them to Alice.
•Alice chooses one of these at random and encrypts it, giving c.
•The adversary chooses a number of ciphertexts (except c) and gets them decrypted by Alice. Finally, the adversary guesses whether m or m` was encrypted.

c. Instead of a subgroup of Z_p one chooses an elliptic curve group of prime order q and with generator g. The secret key x and the random y is chosen in the same way as before, but modular exponentiation is replaced by multiplication with a constant. X, Y and K will be group elements, represented by their coordinates, but this does not prevent $K \| Y$ from being the hash function argument. The expected gain is increased efficiency because of smaller key length.

CHAPTER 11 CRYPTOGRAPHIC HASH FUNCTIONS

11.1 A function h: X →Y is collision resistant if it is computationally infeasible to find two different points x_1, $x_2 \in$ X such that h(x_1) = h(x_2)

11.2 The idea is to split the message to be hashed into blocks of a specified size of, say, *k* bits. Typically messages are padded, using a specified padding scheme, so that the padded message consists of an integral number of full blocks. If the hash function produces digests of size *n* bits, we use a compression function
i.g : $\{0, 1\}^n$ X $\{0, 1\}^k \rightarrow \{0, 1\}^n$.

We further need a specified initialization vector $IV \in \{0, 1\}^n$. Then we iterate the compression function:

ii. $D_0 = IV$

iii. $D_i = g(D_{i-1}, M_i)$

The result of the hash function is the last D_i

11.3 The birthday attack against a hash function h: X →Y is an attack that tries to produce a collision by generating random elements $x \in X$, hashing them and recording the results. The process continues until a collision results. It is essential here that any collision is considered a success; we are not seeking a collision with an a priori fixed *x*. A result from probability theory shows that the expected number of generated elements until a collision occurs is in the order of $\sqrt{N}$, where *N* is the size of *Y*. A common phrasing of this is that an *n*-bit hash function can only provide *n*/2 bits security. An interesting variation is when we generate random elements from two subsets of X and seek a collision such that x_1 is in one of the subsets and x_2 in the other. Again, one can demonstrate that on the order of $\sqrt{N}$ elements from each subset needs to be generated.

11.4 We should expect to get a collision in $O(2^{n/2})$ steps; this is the so-called birthday paradox.

CHAPTER 12 MESSAGE AUTHENTICATION CODES

12.1 We could use triple-DES as a pseudorandom function in order to derive separate keys for the MAC and encryption schemes. Specifically, let *K* be the 168-bit key for triple-DES. Then, compute the MAC-key to be the first 168-bits of (3DES(*K*, 0), 3DES (*K*, 1), 3DES(*K*, 2)) (note that there are 192 bits in the output so just take the first 168). Furthermore, compute the encryption key to be the first 168-bits of (3DES(*K*, 3), 3DES (*K*, 4), 3DES(*K*, 5)). From here on, you can use CBC encryption and a CBC-MAC. (The processor will also need to obtain a random IV for the CBC encryption. We can assume that it has direct access to fresh randomness.) The proof of security works by first replacing the 3DES algorithm with a truly random function. In this case, the derived encryption and MAC keys are uniformly and independently distributed; therefore the MAC and encryption are both secure. The security when using 3DES to derive the keys therefore relies on the assumption that 3DES is a pseudorandom function.

12.2 **a.** Let the `toAccount` of *m* be block M_2 and the adversary's account block M_2'. The adversary then replaces C_2 by

$$C_2' = C_2 \oplus M_2 \oplus M_2' = (M_2 \oplus K_2) \oplus M_2 \oplus M_2') = K_2 \oplus M_2',$$

so when the message $C_1C_2'C_3$ is decrypted it will show the adversary's account as `toAccount`.

b. The message could be authenticated by adding MAC(KM, $M_1M_2M_3$) after the message before encryption. If the receiver checks the MAC before accepting the message, the attack will be discovered.

c. We need to use the key to generate a key stream for xoring with the plaintext. One way to do this is to let K_i = H($K \| i$). Then messages are split into blocks of the size of hash values: m = $M_1M_2...M_n$ and c = $C_1C_2...C_n$ where $C_j = M_j \oplus K_j$. Decryption is the same as encryption (so the receiver generates the same key stream, using the common key K).

12.3 A checksum or CRC, by itself, does not guarantee that the data arriving at the recipient has come from the reported sender. An unauthorized individual can intercept the packet, modify the contents and the checksum, and then forward the modified packet to the recipient.

CHAPTER 13 DIGITAL SIGNATURES

13.1 First the message is hashed and then the signature is applied only to the hash value.

CHAPTER 14 KEY MANAGEMENT AND DISTRIBUTION

14.1 It can be done at follows.
Alice picks a random number r and sends it to Bob.
Alice computes $a'=a \oplus r$ and sends it to Chris.
Bob computes $b'=b \oplus r$ and sends it to Chris.
Chris computes $a' \oplus b' = a \oplus b \oplus (r \oplus r) = a \oplus b$.
Here Chris does not have any info about a, b besides sum. Alice and Bob also have knowledge only about r

14.2 The primary challenge of symmetric encryption algorithms is keeping the single key secure. Known as key management, it poses a number of significant challenges. If a user wants to send an encrypted message to another using symmetric encryption, he must be sure that she has the key to decrypt the message. How should the first user get the key to the second user? He would not want to send it electronically through the Internet, because that would make it vulnerable to eavesdroppers. Nor can he encrypt the key and send it, because the recipient would need some way to decrypt the key. And if he can even get the get securely to the user, how can be he certain that an attacker has not seen the key on that person's computer? Key management is a significant impediment to using symmetric encryption

CHAPTER 15 USER AUTHENTICATION

15.1 **a.** Victor checks that $R \times S = X$ (since $R \times S = g^{r+(x-r)} = g^x = X$) and either $R = g^z$ (if $b = 0$) or $S = g^z$ (if $b = 1$)

b. If the false Peggy guesses that she will get $b = 0$ in message 2, she chooses r at random, and sends $R = g^r$, $S = R^{-1}X$. Her values will then pass Victor's check. If she guesses that $b = 1$, exchange R and S. In both cases, $z = r$.

c. Repeat the protocol t times and accept only if the check succeeds each time. Then a false Peggy has probability 2^{-t} to be accepted

15.2 **a.** In an authenticated key-agreement protocol that uses public key cryptography; perfect forward secrecy (or PFS) is the property that ensures that a session key derived from a set of long-term public and private keys will not be compromised if one of the (long-term) private keys is compromised in the future. Perfect Forward Secrecy works on the premise that no key used for the transfer of data may be used to derive any keys for future transmission.

b. Kerberos does not provide Perfect Forward Secrecy for two reasons. First, it does not use public-key cryptography and so does not fit the full definition; second, because it uses master keys which are long term secrets.

15.3 Yes, an eavesdropping attacker can do off-line password guessing. The attacker has R and X, and does the following:

```
repeat
{
choose candidate password cpw;
compute cJ from cpw;
compute cX ← encrypt(R) with key J
}
until cX = X;
//cpw = pw
```

15.4 Here we give two solutions.

Solution 1 (detailed handshake at end):

After obtaining L', principal A initiates DH to establish a session key S, where the DH messages are encrypted by L'. Encrypting the DH messages by L' ensures that the attacker cannot hijack the data exchange phase (otherwise the attacker can spoof A in the DH and thus have the session key established between itself and B).

Solution 2:
After obtaining L', principal A initiates DH (with unencrypted messages) to establish a session key S. Then B sends a challenge, say R, encrypted with S, to which A responds with a message M applying S and L to R, for example:

- [encrypt(encrypt(R) with L) with S]
- [encrypt(hash(R | L) with S]
- [encrypt(R+1) with S]

CHAPTER 17 TRANSPORT-LEVEL SECURITY

17.1 Here is a brief summary of these terms.

World Wide Web (www) - It is a repository of information spread all over the world and linked together. It uses the concept of interlinked hypertext and hypermedia documents accessed via the Internet. With a browser, a user views web page that may contain text, images, videos, and other multimedia and navigates between them via hyperlinks. The World Wide Web was created in 1989 by Sir Tim Berners-Lee, working at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland and released in 1992.

Hypertext - In hypertext information is stored in a set of documents that are linked using the concept of pointers. The most famous implementation of hypertext is the World Wide Web. Ted Nelson coined the words "hypertext" and "hypermedia" in 1965 and worked with Andries van Dam to develop the Hypertext Editing System in 1968 at Brown University. Hypertext documents can either be static (prepared and stored in advance) or dynamic (continually changing in response to user input). Static hypertext can be used to cross-reference collections of data in documents, software applications, or books on CDs. Hypertext can develop very complex and dynamic systems of linking and cross-referencing.

Hypermedia - It is used as a logical extension of the term hypertext in which graphics, audio, video, plain text and hyperlinks intertwine to create a generally non-linear medium of information. A term first used in a 1965 article by Ted Nelson. Most modern hypermedia is delivered via electronic pages from a variety of systems including Media players, web browsers, and stand-alone applications. Audio hypermedia is emerging with voice command devices and voice browsing.

CHAPTER 18 WIRELESS NETWORK SECURITY

18.1 It could be done via evil twin. In security, an evil twin is a home-made wireless access point that masquerades as a legitimate hot spot to gather personal or corporate information without the end-user's knowledge. It's fairly easy for an attacker to create an evil twin, simply by using a mobile Internet device such as a laptop or a Smartphone and some readily-available software. The attacker positions himself in the vicinity of a legitimate Wi-Fi access point and lets his Internet device discover what name (SSID) and radio frequency the legitimate access point uses. He then sends out his own radio signal, using the same name. To the end-user, the evil twin looks like a hot spot with a very strong signal; that's because the attacker has not only used the same network name and settings as the "good twin" he is impersonating, he has also physically positioned himself near the end-user so that his signal is likely to be the strongest within range. If the end-user is tempted by the strong signal and connects manually to the evil twin to access the Internet, or if the end-user's computer automatically chooses that connection because it is running in promiscuous mode, the evil twin becomes the end-user's Internet access point, giving the attacker the ability to intercept sensitive data such as passwords or credit card information.

CHAPTER 19 ELECTRONIC MAIL SECURITY

19.1 We can do so by TLS or SSL. Transport Layer Security (TLS) and Secure Sockets Layer (SSL), are cryptographic protocols that provide security and data integrity for communications over networks such as the Internet. TLS and SSL encrypt the segments of network connections at the Transport Layer end-to-end.

19.2 Proof of submission is a proof that a message is transmitted to electronic mail system. Non-repudiation enables a recipient of a message to prove in a court of law that it was sent by a particular sender. Non-repudiation in e-commerce prevents initiators of a transaction later claiming that the recipient or some party made the transaction in their name.

CHAPTER 20 IP SECURITY

20.1 There is no definite answers to that but one can form an opinion by first considering what IPsec is and what it does. IPsec refers to a set of standards developed by the Internet Engineering Task Force (IETF). IPsec solves two problems that have plagued the IP protocol suite for years: host-to-host authentication (which will let hosts know that they're talking to the hosts they think they are) and encryption (which will prevent attackers from being able to watch the traffic going between machines).
Neither of these problems is what firewalls were created to solve. Although firewalls can help to mitigate some of the risks present on an Internet without authentication or encryption, there are really two classes of problems here: integrity and privacy of the information flowing between hosts and the limits placed on what kinds of connectivity is allowed between different networks. IPsec addresses the former class and firewalls the latter. Note however from Chapter 19 that IPsec does provide a limited type of firewall capability in that it allows the user to specify traffic processing rules for a variety of classes of traffic. This is a firewall type of service, but IPsec only provides a limited flexibility in this area.

20.2 IPsec would reject the packets and would not pass them to TCP. In SSL, such packets could cause the session to break.

CHAPTER 21 MALICIOUS SOFTWARE

21.1 Bot Net: A peer-to-peer network of compromised hosts controlled by a owner.
Easter Egg: Pretty unspecified code hidden in a program by developers.
Logic Bomb: Code will delete files or crash a system at a certain time

21.2 The major problem is Buffer overrun. If the input string contains a newline character, then this will write past the end of the input buffer. In the worst case, the size of the string might double. For instance, if the caller allocates a buffer on the stack that is just large enough to hold the string, and passes it to escape(), then a stack-smashing attack would be possible.
Another problem is that memcpy() invokes undefined behavior when invoked on overlapping memory regions.

CHAPTER 22 INTRUDERS

22.1 Some of the ways by which hackers' compromise computers without code breaking are as follows.

- Key Catcher (hw or sw)
- Via email that has an executable file for an attachment.
- A boot CD that has its own Operating System

22.2 A Null session problem is commonly a problem that exists on many Systems especially Microsoft based systems where the system allows a person or other system to connect to it without use of username and/or password such as Shares.

22.3 There are many ways to achieve this.

By alerting administrators via email/pager/phone
By changing firewall configurations to
- increase logging of suspect sessions
- block certain sensitive areas inside or
- block offending areas outside
- throttle offending or suspect traffic
- bring down the Internet connection

There are customer filters that can be configured for signatures that an IDS system looks for and there are standard "out of the box" attack signatures that are known attacks. If the IDS is not configured properly it may send what are known as "false positives" or alerts to an over abundance of traffic, therefore overwhelming people with alerts. While such alerts are active responses, they (as stated above) may become overwhelming.

22.4 **a.** Adversary

- Sees the messages: N, R, X
- Computes Hash(R)
- Computes X XOR Hash(R) = Hash(P)

Later on:

- Adversary Requests Login, submits N.
- Machine generates random number R'.
- Adversary computes Hash(R').
- Adversary computes y = Hash(P) XOR Hash(R').
- Adversary submits y, and logs in as user.

b. To strengthen, simply require the protocol to compute
Hash(R XOR P) instead of Hash(R) XOR Hash(P)

CHAPTER 23 FIREWALLS

23.1 No. The phrase "Make money fast" might be spread across multiple packets (e.g., "Make money" in the first packet, "fast" in the second). A stateless packet filter cannot remember any state from prior packets, so cannot usefully block such email.

23.2 Here are some of the threats and their brief explanation.

(1) Attacks against open ports, such as buffer overrun attacks against unblocked services;

(2) Malicious code or attacks carried in email or web traffic (many firewalls do not scan or examine email and web payloads);

(3) Attacks on the firewall itself (e.g., trying to penetrate the firewall code by exploiting a buffer overflow in the firewall's packet parsing code);

(4) Internal attacks by malicious insiders;

(5) Attacks from compromised internal machines against other internal machines (e.g., a laptop becomes infected with a worm, which tries to infect other inside hosts)—applies to perimeter firewalls;

(6) Attacks from compromised machines that have a VPN or other tunnel through the firewall—applies to perimeter firewalls;

(7) Denial of service attacks against the network link or the firewall itself.

23.3 Yes. An ruleset such as the following will do the trick:

```
drop tcp *:* -> 5.6.7.8:*
```

The following might be a little better, because it does not restrict outbound connections initiated by our internal server:

```
drop tcp *:* -> 5.6.7.8:* (if SYN flag set)
```

23.4 **a.** Here are the strengths.

(1) It mediates all incoming traffic from external hosts and can protect against many attacks by outsiders;

(2) It is easier to manage and to update policies, because of single central location;

(3) It protects against some kinds of DoS attacks launched from the outside.

Here are the weaknesses.

(1) It has no protection against malicious insiders;

(2) It has no protection for mobile laptops while they are connected to other networks;

(3) It has no protection if laptops get infected while travelling and then spread infection when they re-connect to our internal network

b. Here are the strengths.
(1) It protects against malicious insiders and infected internal machines as well as outside attackers;
(2) It protects laptops even while they are travelling and connected to other networks;
(3) It may be easier to customize firewall protection on a per-machine basis.

Here are the weaknesses.
(1) It is potentially more difficult to manage policies, due to the number of machines whose rulesets must be configured and updated;
(2) Uncooperative users may be able to modify settings or disable firewalls on their own machines, and viruses/worms may be able to do the same to machines they infect;
(3) It is potentially less resistant to DDoS, since DoS attacks can still flood internal network links;
(4) Depending upon firewall configuration, it may block legitimate internal traffic and/or make some internal services harder to use.

c. Here are the strengths.
(1) Layered defense provides redundancy in case one firewall fails;
(2) It can easily update policy against external attacks if a new threat develops, which gives some time to update the rulesets on internal hosts.
(3) Strengths (a)(1) and (b)(1)–(3) also apply.

Here are the weaknesses.
(1) Potential for overblocking of legitimate traffic, since traffic flows only if permitted by both firewalls.
(2) Weaknesses (b)(1), (b)(4) also apply