SOLUTIONS MANUAL

CRYPTOGRAPHY AND NETWORK SECURITY: PRINCIPLES AND PRACTICE EIGHTH EDITION

CHAPTERS 11–23

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 11 CRYPTOGRAPHIC HASH FUNCTIONS

ANSWERS TO QUESTIONS

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

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

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

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

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

11.6 The criteria fall into three categories:
•**Security**: The evaluation considered the relative security of the candidates compared to each other and to SHA-2. In addition, specific security requirements related to various applications and resistance to attacks are included in this category.
•**Cost**: NIST intends SHA-3 to be practical in a wide range of applications. Accordingly, SHA-3 must have high computational efficiency, so as to be usable in high-speed applications, such as broadband links, and low memory requirements.
•**Algorithm and implementation characteristics**: This category includes a variety of considerations, including flexibility; suitability for

a variety of hardware and software implementations; and simplicity, which will make an analysis of security more straightforward.

11.7 The sponge construction has the same general structure as other iterated hash functions. The sponge function takes an input message and partitions it into fixed-size blocks. Each block is processed in turn with the output of each iteration fed into the next iteration, finally producing an output block. Unlike a typical hash function, the sponge construction allows for a variable-length output.

11.8 The function *f* is executed once for each input block of the message to be hashed. The function takes as input the 1600-bit state variable and converts it into a 5×5 matrix of 64-bit lanes. This matrix then passes through 24 rounds of processing. Each round consists of five steps, and each step updates the state matrix by permutation or substitution operations. The rounds are identical with the exception of the final step in each round, which is modified by a round constant that differs for each round.

11.9 **Theta:** New value of each bit in each word depends its current value and on one bit in each word of preceding column and one bit of each word in succeeding column.
Pho: The bits of each word are permuted using a circular bit shift. W[0, 0] is not affected.
Pi: Words are permuted in the 5×5 matrix. W[0, 0] is not affected.
Chi: New value of each bit in each word depends on its current value and on one bit in next word in the same row and one bit in the second next word in the same row.
Theta: W[0, 0] is updated by XOR with a round constant.

ANSWERS TO PROBLEMS

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

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

error bits in each parity column and an even number of error bits in each spiral.

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

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

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

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

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

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

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

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

c. 229

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

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

For DES, the function f is depicted in Figure 3.6. It maps a 32-bit R and a 48-bit K into a 32-bit output. That is, it maps an 80-bit input into a 32-bit output. This is clearly a one-way function. Any hash function that produces a 32-bit output could be used for f. The demonstration in the text that decryption works is still valid for any one-way function f.

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

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

then

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

so

$$RSAH(C1, C2) = RSA[RSA(C1) \oplus C2)] = RSA[RSA(B1) \oplus B2]$$
$$= RSAH(B1, B2)$$

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

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

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

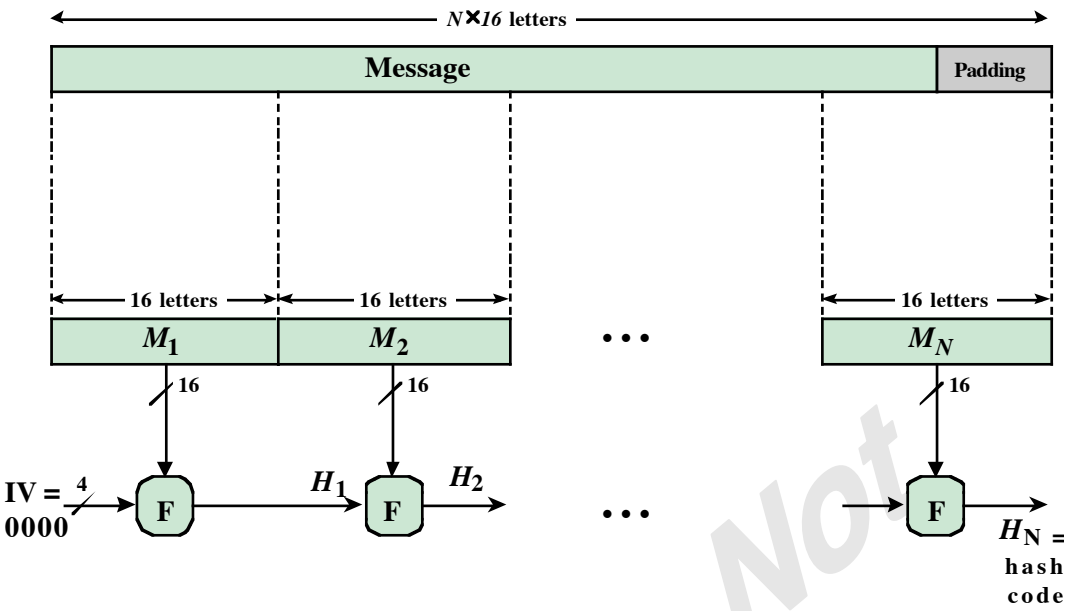
11.8 $W_{16} = W_0 \oplus \sigma_0(W_1) \oplus W_9 \oplus \sigma_1(W_{14})$
$W_{17} = W_1 \oplus \sigma_0(W_2) \oplus W_{10} \oplus \sigma_1(W_{15})$
$W_{18} = W_2 \oplus \sigma_0(W_3) \oplus W_{11} \oplus \sigma_1(W_{16})$
$W_{19} = W_3 \oplus \sigma_0(W_4) \oplus W_{12} \oplus \sigma_1(W_{17})$

11.9 **a.** 1 bit
b. 1024 bits
c. 1023 bits

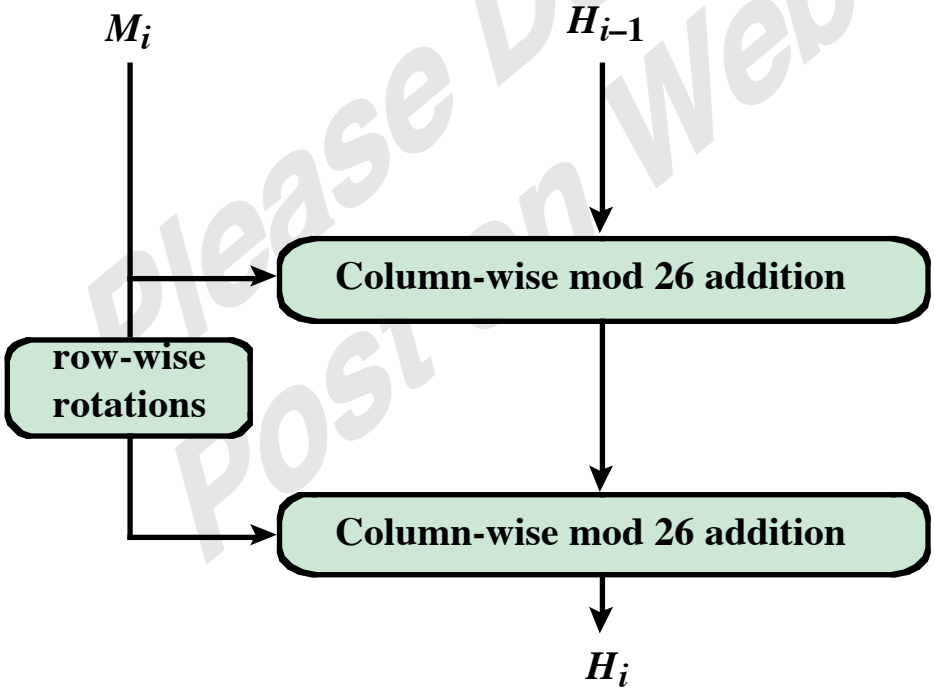
11.10 **a.** 1919
b. 1920
c. 1921

11.11 **a.** **1.** Interchange x_1 and x_4; x_2 and x_3; y_1 and y_4; and y_2 and y_3.
2. Compute $Z = X + Y \bmod 2^{32}$.
3. Interchange z_1 and z_4; and z_2 and z_3.
b. You must use the same sort of interchange.

11.12 **a.** Overall structure:



Compression function F:



b. BFQG

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

AYHGDAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA

11.13 $c = 448$: 448/64 = 7 lanes of all zeros. This includes all 5 lanes in row $y = 0$, plus two lanes in row $y = 1$, namely $L[0, 1]$, $L[1, 1]$.
$c = 512$: 512/64 = 8 lanes of all zeros. This includes all the lanes in row $y = 0$, plus three lanes in row $y = 1$, namely $L[0, 1]$, $L[1, 1]$, $L[2, 1]$.
$c = 768$: 768/64 = 12 lanes of all zeros. This includes all the lanes in rows $y = 0$ and y = 1, plus two lanes in row $y = 2$, namely $L[0, 2]$, $L[1, 2]$.
$c = 1024$: 1024/64 = 16 lanes of all zeros. This includes all the lanes in rows $y = 0$, $y = 1$, and $y = 2$, plus $L[0, 3]$.

11.14 Potentially, all of the lanes will have at least one 1 bit after the theta step function in Round 0. This is because every column has at least one nonzero lane, and every lane is updated by the XOR of itself and all of the lanes in the preceding and following columns (with a bit position shift in the following column). It is possible that the XOR would result in a zero result for all 64 bits of a lane and so that lane would remain zero. In that case the chi step function is the next possible function to achieve the result we are looking for. In this case, a lane is updated as a function of the next two lanes in its row. If we assume that a given lane is all zeros, then the calculation is (see Equation 11.4)

$$\text{NOT}(a[x+1]) \text{ AND } a[x+2]$$

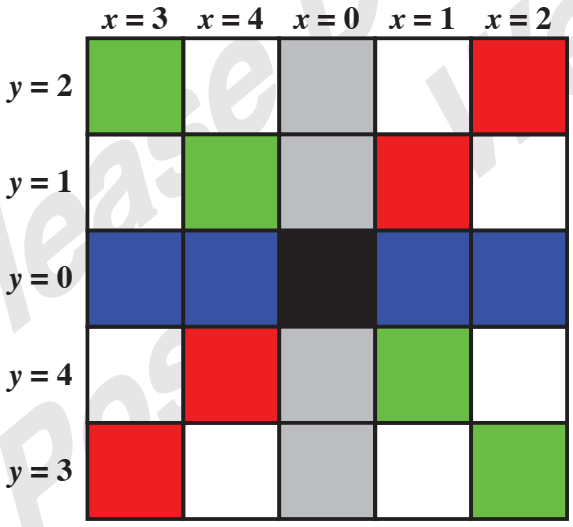
This will yield a zero result unless there is at least one bit position out of 64 bit positions such that that bit is 0 in the (x+1) lane and 1 in the (x+2). If at the end of Round 0 there is still one or more of the initial zero words that are still zero, then there is a high probability that it will pick up at least one 1 bit in the theta or chi step of Round 1.

11.15 It is perhaps easier to visualize the permutation in this orientation. We have:

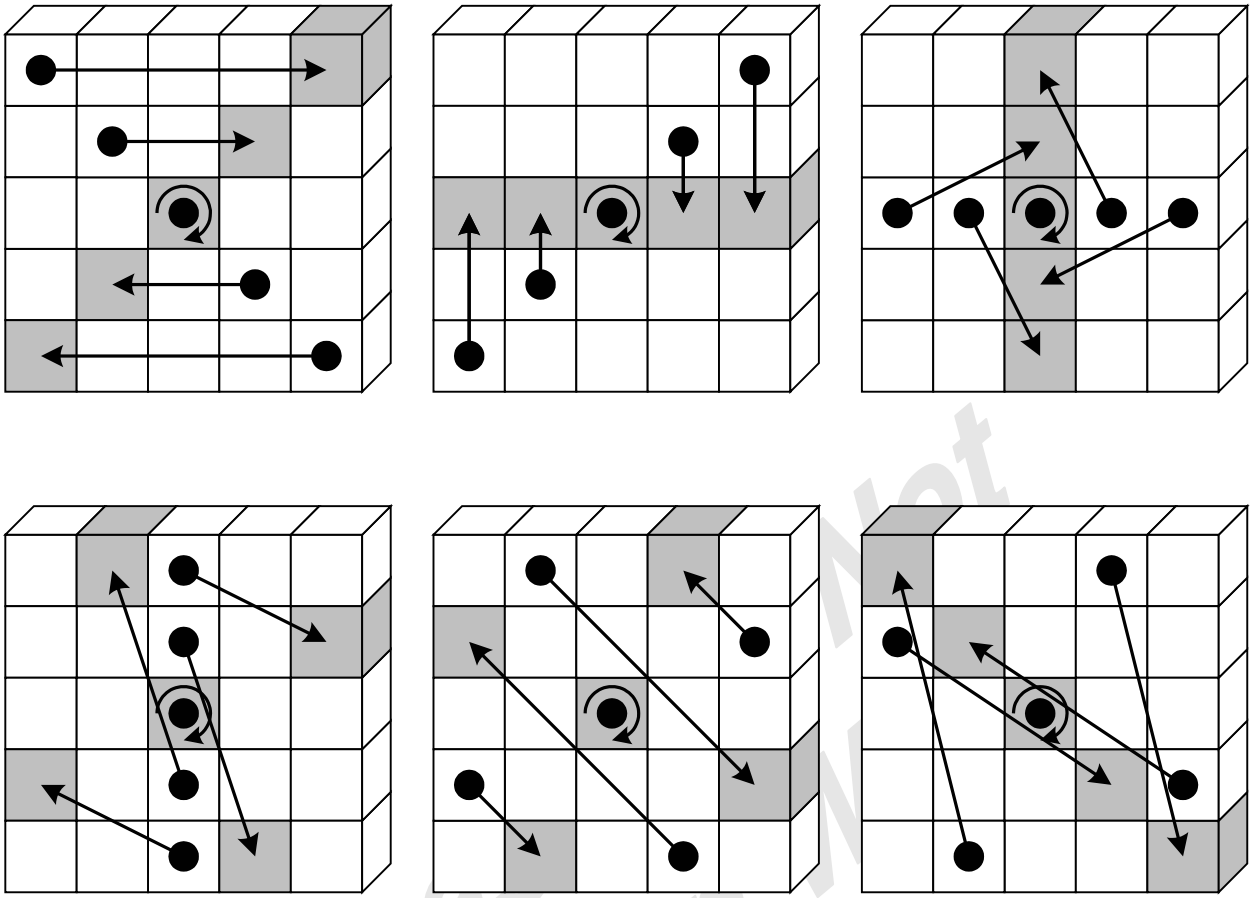
	$x=3$	$x=4$	$x=0$	$x=1$	$x=2$
$y=2$	L[3,2]	L[4,2]	L[0,2]	L[1,2]	L[2,2]
$y=1$	L[3,1]	L[4,1]	L[0,1]	L[1,1]	L[2,1]
$y=0$	L[3,0]	L[4,0]	L[0,0]	L[1,0]	L[2,0]
$y=4$	L[3,4]	L[4,4]	L[0,4]	L[1,4]	L[2,4]
$y=3$	L[3,3]	L[4,3]	L[0,3]	L[1,3]	L[2,3]

→

	$x=3$	$x=4$	$x=0$	$x=1$	$x=2$
$y=2$	L[4,3]	L[0,4]	L[1,0]	L[2,1]	L[3,2]
$y=1$	L[1,3]	L[2,4]	L[3,0]	L[4,1]	L[0,2]
$y=0$	L[3,3]	L[4,4]	L[0,0]	L[1,1]	L[2,2]
$y=4$	L[0,3]	L[1,4]	L[2,0]	L[3,1]	L[4,2]
$y=3$	L[2,3]	L[3,4]	L[4,0]	L[0,1]	L[1,2]



Now visualize the matrix as having 4 straight lines through five squares: vertical, horizontal, and the two diagonals. The center square of each line is L[0,0], which does not change position. Now consider this figure from the Keccak documentation show the pi permutation when the rows and columns are organized as indicated above.



We see that the falling diagonal is mapped to the rising diagonal, the rising diagonal is mapped to the central row, and the central row is mapped into the central column. The remaining three mappings are less succinctly described. Still, this orientation is useful for getting a feel for what the permutation accomplishes.

11.16 Let us say we are concerned with the execution of the iota function in round i.

a. During the theta step function in round i+1, every lane in column $x = 1$ and column $x = 4$ is updated with $L[0, 0]$ as one of the inputs to the calculation. For a moment, ignore the pi step and assume that we were to go immediately to the chi step. Every lane in column x = 2 and x = 3 is affected by the corresponding lane in x = 4, which has already been updated in the theta step to incorporate $L[0, 0]$. Similarly, every lane in x = 0 is affected by the corresponding lane in x = 1, which has already been updated in the theta step to incorporate $L[0, 0]$. Thus, during round i+1 all lanes are affected by the changes to $L[0, 0]$ during round i, via the theta and step functions. It can be shown that the permutation pi does not affect this reasoning. This is left as a further exercise to the student.

b. Keep in mind that only a few bit positions in $L[0, 0]$ are affected by the iota function (at most 6). Thus during round $i+1$, only a few bit positions in each lane are affected. By the same reasoning as that of the answer to Problem 11.14, we can expect that there is high probability that all bit will be affected by the end of round $i+2$, and even higher probability by the end of round $i+3$.

CHAPTER 12 MESSAGE AUTHENTICATION CODES

ANSWERS TO QUESTIONS

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

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

12.3 Message encryption, message authentication code, hash function.

12.4 Error control code, then encryption.

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

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

12.7 Figures 11.2 and 11.3 illustrate a variety of ways in which a hash code can be used to provide message authentication, as follows. **Figure**

11.2: a. The message plus concatenated hash code is encrypted using symmetric encryption. **b.** Only the hash code is encrypted, using symmetric encryption. **c.** Only the hash code is encrypted, using public-key encryption and using the sender's private key. **d.** If confidentiality as well as a digital signature is desired, then the message plus the public-key-encrypted hash code can be encrypted using a symmetric secret key. **Figure 11.3: a.** This technique uses a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify. **b.** Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

12.8 No. Section 12.4 outlines such attacks.

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

ANSWERS TO PROBLEMS

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

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

12.3 We use the definition from Section 12.6. For a one-block message, the MAC using CBC-MAC is $T = E(K, X)$, where K is the key and X is the message block. Now consider the two-block message in which the first block is X and the second block is $X \oplus T$. Then the MAC is $E(K, [T \oplus \square X \oplus T\square]) = E(K, X) = T$.

12.4 We use Figure 12.8a but put the XOR with K_1 after the final encryption. For this problem, there are two blocks to process. The output of the encryption of the first message block is $E(K, \mathbf{0}) = CBC(K, \mathbf{0}) = T_0 \oplus K_1$. This is XORed with the second message block ($T_0 \oplus T_1$), so that the input to the second encryption is $(T_1 \oplus K_1) = CBC(K, \mathbf{1}) = E(K, \mathbf{1})$. So the output of the second encryption is $E(K, [E(K, \mathbf{1})]) = CBC(K, [CBC(K, \mathbf{1})]) = T_2 \oplus K_1$. After the final XOR with K_1, we get

VMAC(K, [$\mathbf{0} \| (T_0 \oplus T_1)$]) = T_2.

12.5 a. In each case (64 bits, 128 bits) the constant is the binary representation of the irreducible polynomial defined in Section 12.6. The two constants are

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

b. Here is the algorithm from the NIST document:

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

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

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

12.6 a. MtE

b. This is basically the E&M approach, but uses one key instead of two.

12.7 As was discussed in Chapter 5, multiplication distributes over addition in a field, and for this type of field the XOR operation is the addition operation. Consider a message consisting of two blocks. Then by Figure 12.10a, the GHASH function is

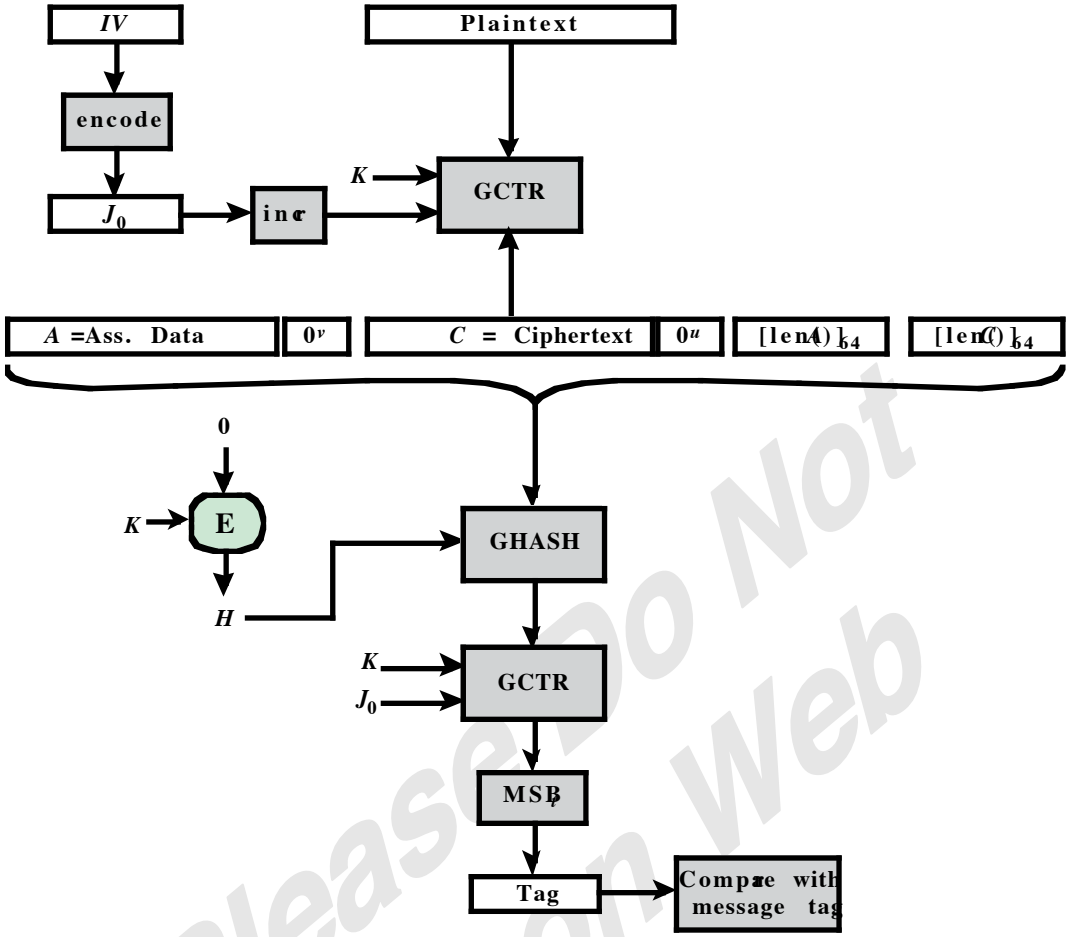
$$(X_1 \bullet H) \oplus X_2) \bullet H)$$

Multiplying through by H, we get the expression

$$(X_1 \bullet H^2) \oplus (X_2 \bullet H)$$

This calculation can be extended through more blocks, up to X_m, to get the expression shown in the statement of this problem.

12.8



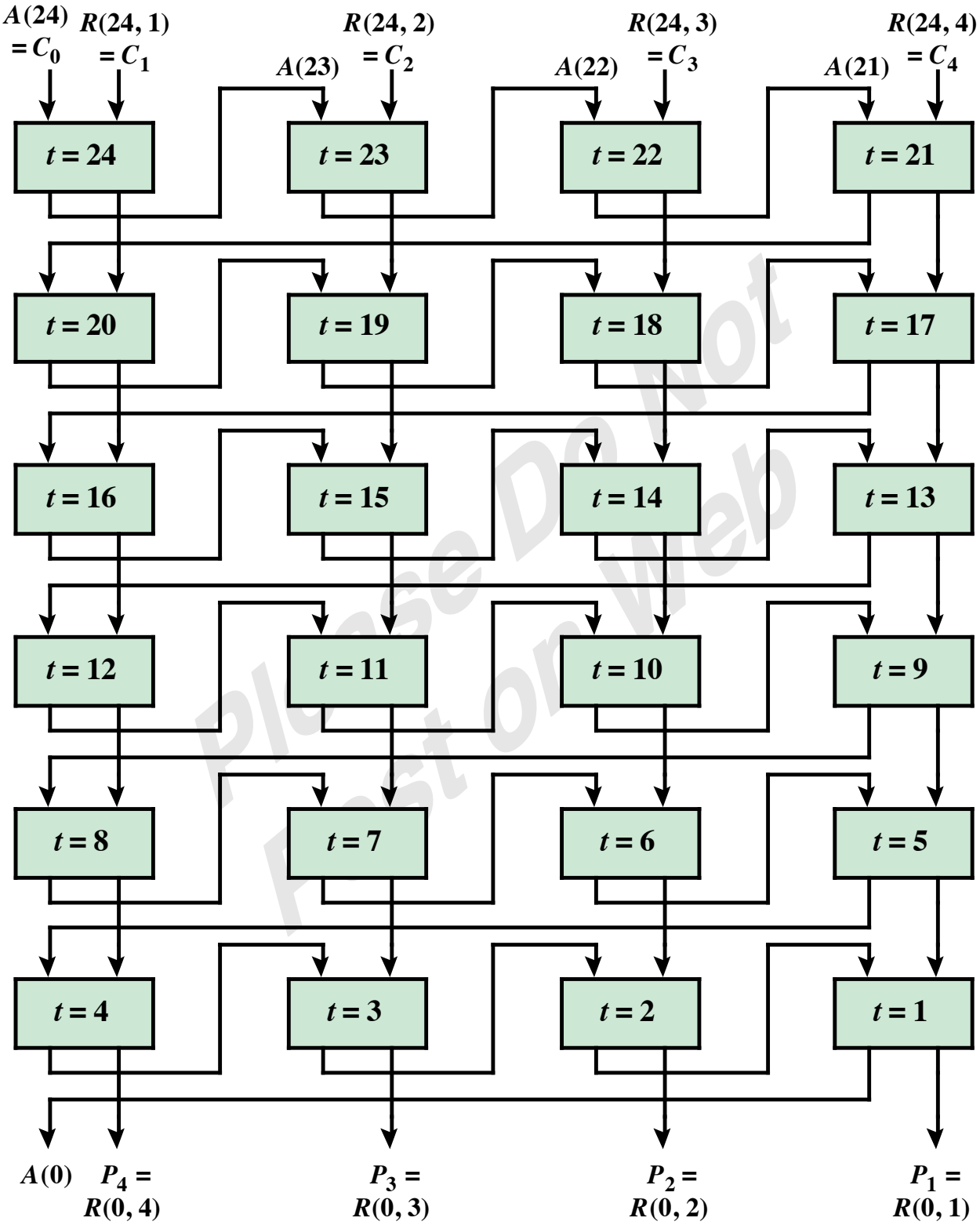
12.9 a. The following matrix shows the message for each received 2-bit word.

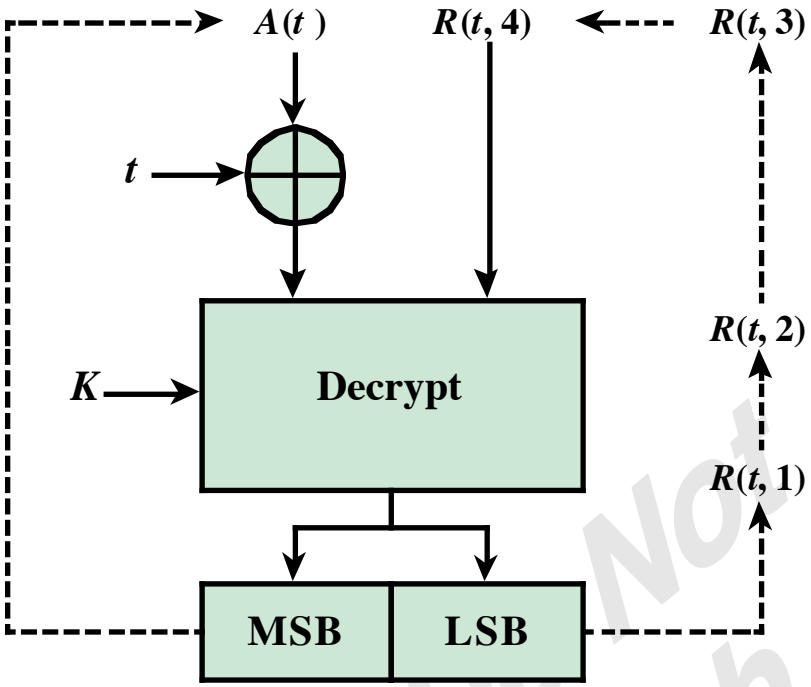
	Word			
Key	00	01	10	11
1	0	1	—	—
2	1	—	0	—
3	—	0	—	1
4	—	—	1	0

b. The probability that some one can successfully impersonate Alice is 0.5 because only two of the four words are possible as transmitted word under the joint secret key.

c. An opponent Eve who tries to replace a transmitted message by another one will know that only two keys can possibly have been used, but she doesn't know which one. So, the probability of a successful substitution is also 0.5.

12.10





12.11 a. This is functionally equivalent; that is, it produces the same result. This alternative description of the key wrap algorithm involves indexing rather than shifting. This approach allows one to calculate the wrapped key in place, avoiding the rotation in the algorithm in Section 12.8. This produces identical results and is more easily implemented in software.

b.

1. Initialize variables.

$A = C_0$

for $i = 1$ **to** n

$R(i) = C_i$

2. Calculate intermediate values.

for $j = 5$ **to** 0

for $i = n$ **to** 1

$t = (n \times j)+i$

$B = \text{E}(K, [(A \oplus t) \,||\, R(i)])$

$A = \text{MSB}_{64}(B)$

$R(i) = \text{LSB}_{64}(B)$

3. Output results.

if $A(0)$ = A6A6A6A6A6A6A6A6

then

for $i = 1$ **to** n

$P(i) = R(i)$

else

return error

CHAPTER 13 DIGITAL SIGNATURES

ANSWERS TO QUESTIONS

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

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

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

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

13.5 It is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to

the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

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

ANSWERS TO PROBLEMS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

13.8 a. The receiver validates the digital signature by ensuring that the first 56-bit key in the signature will encipher validation parameter *u*1 into E(*k*1, *u*1) if the first bit of *M* is 0, or that it will encipher *U*1 into E(*K*1, *U*1) if the first bit of *M* is 1; the second 56-bit key in the signature will encipher validation parameter *u*2 into E(*k*2, *u*2) if the second bit of *M* is 0, or it will encipher *U*2 into E(*K*2, *U*2) if the second bit of *M* is 1,; and so on.

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

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

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

CHAPTER 14 LIGHTWEIGHT CRYPTOGRAPHY AND POST-QUANTUM CRYPTOGRAPHY

ANSWERS TO QUESTIONS

14.1 The term *embedded system* refers to the use of electronics and software within a product that has a specific function or set of functions, as opposed to a general-purpose computer, such as a laptop or desktop system. We can also define an embedded system as any device that includes a computer chip, but that is not a general-purpose workstation, desktop or laptop computer.

14.2 A constrained device is a device with limited volatile and nonvolatile memory, limited processing power, and a low data rate transceiver.

14.3 **Class 0**: These are very constrained devices, typically sensors, called motes, or smart dust. Motes can be implanted or scattered over a region to collect data and pass it on from one to another to some central collection point. Class 0 devices generally cannot be secured or managed comprehensively in the traditional sense. They will most likely be preconfigured (and will be reconfigured rarely, if at all) with a very small data set.

Class 1: These are quite constrained in code space and processing capabilities, such that they cannot easily talk to other Internet nodes employing a full protocol stack. However, they are capable enough to use a protocol stack specifically designed for constrained nodes and participate in meaningful conversations without the help of a gateway node.

Class 2: These are less constrained and fundamentally capable of supporting most of the same protocol stacks as used on notebooks or servers. However, they are still very constrained compared to high-end IoT devices. Thus, they require lightweight and energy-efficient protocols and low transmission traffic.

14.4 **Chip area**: Chip area is of concern when a cryptographic algorithm is implemented in hardware. Very small devices, such as small sensors, have limited available chip area to provide for security.

Energy consumption: Many constrained devices operate from a very small battery or from energy derived from an incoming signal.

Accordingly, algorithms may need to be designed to minimize energy consumption.

Program code size and RAM size: Constrained devices typically have very limited space for program code (e.g., in ROM) and RAM needed for execution. Thus, cryptographic algorithms need to be compact in terms of code and make use of minimal RAM during execution.

Communications transmission rate: Very constrained devices, such as sensors and RFID tags, may be capable of very limited data rates. Thus, the amount of security-related data that needs to be transmitted, such as message authentication codes and key exchange material, needs to be extremely small.

Execution time: For some devices, such as contactless cards and RFID tags, execution time is constrained by the amount of time the device is present in the communication zone.

14.5 •Many iterations of simple rounds
•Simple operations like XORs, rotation, 4□□ 4 S-boxes, and bit permutations
•Smaller block sizes (e.g., 64 or 80 bits)
•Smaller key sizes (e.g., 96 or 112 bits)
•Simpler key schedules
•Small security margins by design
•Many iterations of simple rounds
•Simplified key schedules that can generate sub-keys on the fly

14.6 Work on lightweight cryptography is almost exclusively devoted to symmetric (secret key) algorithms and cryptographic hash functions.

14.7 Post-quantum cryptography is an area of study that arises from the concern that quantum computers would be able to break currently-used asymmetric cryptographic algorithms.

14.8 Work on post-quantum cryptography is devoted to developing new asymmetric cryptographic algorithms.

14.9 ●Lattice-Based Cryptographic Algorithms
●Code-Based Cryptographic Algorithms
●Multivariate-Based Cryptographic Algorithms
●Hash-Based Digital Signature Algorithms

CHAPTER 15 KEY MANAGEMENT AND DISTRIBUTION

ANSWERS TO QUESTIONS

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

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

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

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

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

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

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

updates could be listed in a widely circulated newspaper. **5.** Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

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

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

15.9 X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.

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

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

ANSWERS TO PROBLEMS

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

b. The proposed scheme appears to provide the same degree of security as that of Figure 15.3. One advantage of the proposed scheme is that the, in the event that B rejects a connection, the overhead of an interaction with the KDC is avoided.

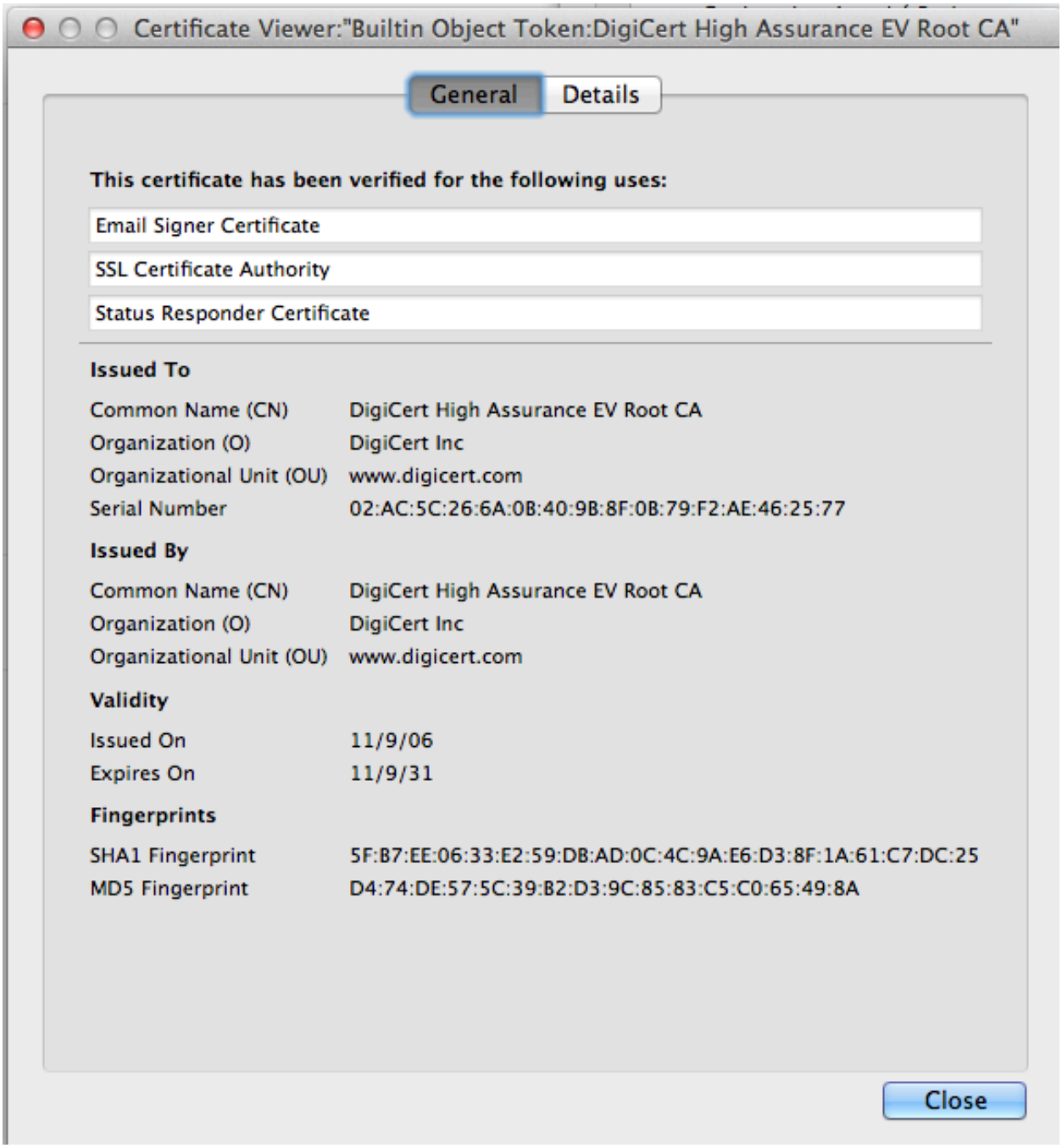
15.2 i) sending to the server the source name A, the destination name Z (his own), and E(K_a, R), as if A wanted to send him the same message encrypted under the same key R as A did it with B

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

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

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

15.4 Here is an example of a trusted root CA certificate from Firefox.



15.5 When a symmetric key is used to protect stored information, the recipient usage period may start after the beginning of the originator usage period as shown in the figure. For example, information may be encrypted before being stored on a compact disk. At some later time, the key may be distributed in order to decrypt and recover the information.

15.6 a. A believes that she shares K'_{AB} with B since her nonce came back in message 2 encrypted with a key known only to B (and A). B believes that he shares K'_{AB} with A since N_A was encrypted with K'_{AB}, which could only be retrieved from message 2 by someone who knows K'_{AB} (and this is known only by A and B). A believes that K'_{AB} is fresh since it is included in message 2 together with N_A (and hence message 2 must have been constructed after message 1 was sent). B believes (indeed, knows) that K'_{AB} is fresh since he chose it himself.

b. B. We consider the following interleaved runs of the protocol:

1. A $\rightarrow$C(B) : A, N_A
1`. C(B) $\rightarrow$A : B, N_A
2`. A $\rightarrow$C(B) : $\mathrm{E}(K_{AB}, [N_A, K'_{AB}])$
2. C(B) $\rightarrow$A : $\mathrm{E}(K_{AB}, [N_A, K'_{AB}])$
3. A $\rightarrow$C(B) : $\mathrm{E}(K'_{AB}, N_A)$

C cannot encrypt A's nonce, so he needs to get help with message 2. He therefore starts a new run with A, letting A do the encryption and reflecting the reply back. A will accept the unprimed protocol run and believe that B is present.

c. To prevent the attack, we need to be more explicit in the messages, e.g. by changing message 2 to include the sender and receiver (in this order), i.e. to be $\mathrm{E}(K_{AB}, [\mathrm{A}, \mathrm{B}, N_A, K'_{AB}])$.

15.7 A typical PKI consists of seven core components. These are briefly described below:

1. Digital certificates (public-key certificates, X.509 certificates): A digital certificate is a signed data structure that binds one or more attributes of an entity with its corresponding public key. By being signed by a recognized and trusted authority (i.e. the Certification Authority) a digital certificate provides assurances that a particular public key belongs to a specific entity (and that the entity possesses the corresponding private key).

2. Certification Authority (CA): Certification Authorities are the people, processes and tools that are responsible for the creation, issue and management of public-key certificates that are used within a PKI.

3. Registration Authority (RA): Registration Authorities are the people, processes and tools that are responsible for authenticating the identity of new entities (users or computing devices) that require certificates from CAs. RAs additionally maintain local registration data and initiate renewal or revocation processes for old or redundant certificates. They

act as agents of CAs (and in that regard can carry out some of the functions of a CA if required).
4. Certificate repository: A database, or other store, which is accessible to all users of a PKI, within which public-key certificates, certificate revocation information and policy information can be held.
5. PKI client software: Client-side software is required to ensure PKI-entities are able to make use of the key and digital certificate management services of a PKI (e.g. key creation, automatic key update and refreshment).
6. PKI-enabled applications: Software applications must be PKI-enabled before they can be used within a PKI. Typically this involves modifying an application so that it can understand and make use of digital certificates (e.g. to authenticate a remote user and authenticate itself to a remote user).
7. Policy (Certificate Policy and Certification Practice Statement): Certificate Policies and Certification Practice Statements are policy documents that define the procedures and practices to be employed in the use, administration and management of certificates within a PKI.

15.8 The primary weakness 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.

15.9 Adding EMK_0 would allow users to generate personal session keys, which could be exchanged, avoiding the necessity of storing a key variable in a user-to-user session.

15.10 Host *i* has master key KMH_i, with variants $KMH_{i,j}$, j = 0, 1, 2.

$KMH_{i,0}$: used to encrypt session key *KS*
$KMH_{i,1}$: used to encrypt user master keys (at Host *i*)
$KMH_{i,2}$: used to encrypt cross domain key KMH(*i*, *j*) = KMH(*j*, *i*)
(Host *i* to Host *j*)

Host *i* stores $E[KMH_{i,2}, KMH(i, j)]$ and uses a translation instruction RFMK':

$$RFMK'[E[KMH_{i,2}, KMH(i, j)], E(KMH_{i,0}, KS)] \rightarrow E(KMH_{i,j}, K)]$$

A second translation function RTMK (at Host *j*)

$$RTMK[E[KMH_{j,2}, KMH(j, i)], E(KMH(i, j), KS)] \rightarrow E(KMH_{j,0}, KS)]$$

which may be deciphered by a user at Host *j*.

15.11 One solution is to add an instruction similar to RFMK of the form

$$KEYGEN[RN, KMT_i, KMT_j]$$

which will interpret RN as E(KMH0, KS) and return both $E(KMH_i, KS)$ and $E(KMH_j, KS)$, which are sent to the terminals *i* and *j*, respectively. RN need not be maintained at the host.

CHAPTER 16 USER AUTHENTICATION

ANSWERS TO QUESTIONS

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

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

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

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

16.5 **1.** A user may gain access to a particular workstation and pretend to be another user operating from that workstation. **2.** A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated

workstation. **3.** A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

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

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

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

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

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

ANSWERS TO PROBLEMS

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

16.2

(1) A → B: $ID_A \| N_a$

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

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

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

(5) A → B: $E(K_s, N_b)$

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

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

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

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

A → B: A $\{r_B, B\}$

16.6 a. This is a means of authenticating A to B. R_1 serves as a challenge, and only A is able to encrypt R_1 so that it can be decrypted with A's public key.

b. Someone (e.g., C) can use this mechanism to get A to sign a message. Then, C will present this signature to D along with the

message, claiming it was sent by A. This is a problem if A uses its public/private key for both authentication, signatures, etc.

16.7 a. This is a means of authenticating A to B. Only A can decrypt the second message, to recover R_2.

b. Someone (e.g. C) can use this mechanism to get A to decrypt a message (i.e., send that message as R_2) that it has eavesdropped from the network (originally sent to A).

16.8 It contains the Alice's ID, Bob's name, and timestamp encrypted by the KDC-Bob secret key.

16.9 It contains Alice's name encrypted by the KDC-Bob secret key.

16.10 It has a nonce (e.g., time stamp) encrypted with the session key.

16.11 It contains the session key encrypted by the KDC-Bob secret key.

CHAPTER 17 TRANSPORT-LEVEL SECURITY

ANSWERS TO QUESTIONS

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

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

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

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

17.5 **Server and client random:** Byte sequences that are chosen by the server and client for each connection. **Server write MAC secret:** The secret key used in MAC operations on data sent by the server. **Client**

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

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

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

17.8 HTTPS (HTTP over TLS) refers to the combination of HTTP and TLS to implement secure communication between a Web browser and a Web server.

17.9 The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security. SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail.

17.10 **Transport Layer Protocol**: Provides server authentication, data confidentiality, and data integrity with forward secrecy (i.e., if a key is compromised during one session, the knowledge does not affect the security of earlier sessions). The transport layer may optionally provide compression. **User Authentication Protocol**: Authenticates the user to the server. **Connection Protocol:** Multiplexes multiple logical communications channels over a single underlying SSH connection.

ANSWERS TO PROBLEMS

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

17.2 To integrity protect the first set of messages where the cookies and crypto suite information is exchanged. This will prevent a man-in-the-middle attack in step 1 for instance, where someone can suppress the original message and send a weaker set of crypto suites.

17.3
a. Brute Force Cryptanalytic Attack: The conventional encryption algorithms use key lengths ranging from 40 to 168 bits.
b. Known Plaintext Dictionary Attack: TLS protects against this attack by not really using a 40-bit key, but an effective key of 128 bits. The rest of the key is constructed from data that is disclosed in the Hello messages. As a result the dictionary must be long enough to accommodate 2^{128} entries.
c. Replay Attack: This is prevented by the use of nonces.
d. Man-in-the-Middle Attack: This is prevented by the use of public-key certificates to authenticate the correspondents.
e. Password Sniffing: User data is encrypted.
f. IP Spoofing: The spoofer must be in possession of the secret key as well as the forged IP address.
g. IP Hijacking: Again, encryption protects against this attack.
h. SYN Flooding: TLS provides no protection against this attack.

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

17.5 This allows for the message to be authenticated before attempting decryption, which may be more efficient.

CHAPTER 18 WIRELESS NETWORK SECURITY

ANSWERS TO QUESTIONS

18.1 Basic service set.

18.2 Two or more basic service sets interconnected by a distribution system.

18.3 **Association:** Establishes an initial association between a station and an AP. **Authentication:** Used to establish the identity of stations to each other. **Deauthentication:** This service is invoked whenever an existing authentication is to be terminated. **Disassociation:** A notification from either a station or an AP that an existing association is terminated. A station should give this notification before leaving an ESS or shutting down. **Distribution:** used by stations to exchange MAC frames when the frame must traverse the DS to get from a station in one BSS to a station in another BSS. **Integration:** enables transfer of data between a station on an IEEE 802.11 LAN and a station on an integrated IEEE 802.x LAN. **MSDU delivery:** delivery of MAC service data units. **Privacy:** Used to prevent the contents of messages from being read by other than the intended recipient. **Reassocation:** Enables an established association to be transferred from one AP to another, allowing a mobile station to move from one BSS to another.

18.4 It may or may not be.

18.5 **Mobility** refers to the types of physical transitions that can be made by a mobile node within an 802.11 environment (no transition, movement from one BSS to another within an ESS, movement from one ESS to another). **Association** is a service that allows a mobile node that has made a transition to identify itself to the AP within a BSS so that the node can participate in data exchanges with other mobile nodes.

18.6 IEEE 802.11i addresses three main security areas: authentication, key management, and data transfer privacy.

18.7 **Discovery:** An AP uses messages called Beacons and Probe Responses to advertise its IEEE 802.11i security policy. The STA uses these to identify an AP for a WLAN with which it wishes to communicate. The STA associates with the AP, which it uses to select the cipher suite and authentication mechanism when the Beacons and Probe Responses present a choice. **Authentication**: During this phase, the STA and AS prove their identities to each other. The AP blocks non-authentication traffic between the STA and AS until the authentication transaction is successful. The AP does not participate in the authentication transaction other than forwarding traffic between the STA and AS. **Key generation and distribution**: The AP and the STA perform several operations that cause cryptographic keys to be generated and placed on the AP and the STA. Frames are exchanged between the AP and STA only **Protected data transfer**: Frames are exchanged between the STA and the end station through the AP. As denoted by the shading and the encryption module icon, secure data transfer occurs between the STA and the AP only; security is not provided end-to-end. **Connection termination**: The AP and STA exchange frames. During this phase, the secure connection is torn down and the connection is restored to the original state.

18.8 TKIP is designed to require only software changes to devices that are implemented with the older wireless LAN security approach called Wired Equivalent Privacy (WEP).

ANSWERS TO PROBLEMS

18.1 **a.** This scheme is extremely simple and easy to implement. It does protect against very simple attacks using an off-the-shelf Wi-Fi LAN card, and against accidental connection to the wrong network.

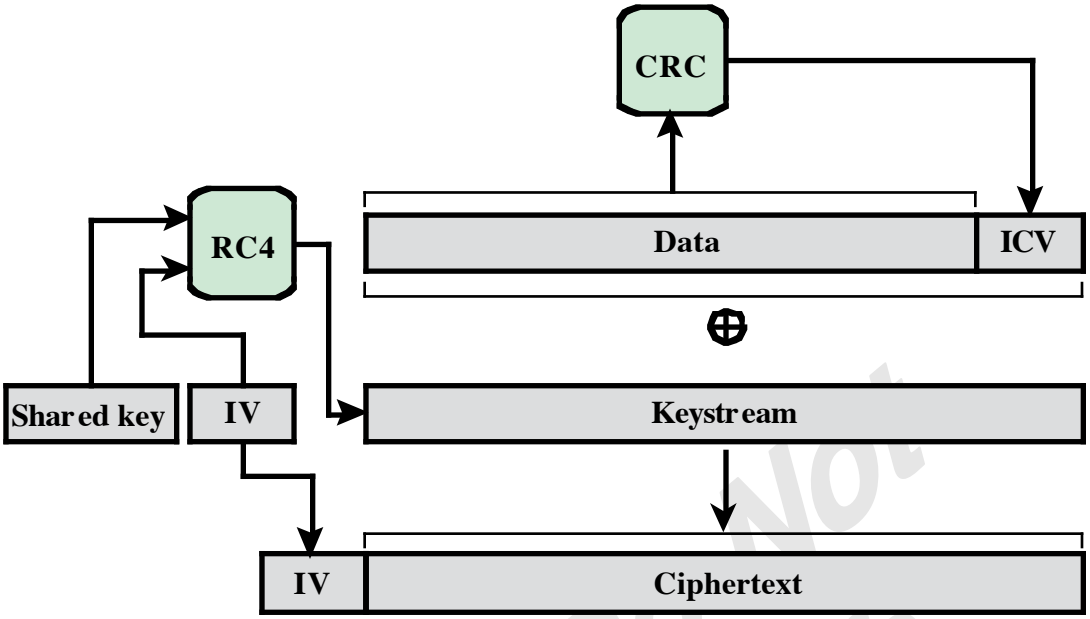
b. This scheme depends on all parties behaving honestly. The scheme does not protect against MAC address forgery.

18.2 **a.** Because the AP remembers the random number previously sent, it can check whether the result sent back was encrypted with the correct key; the STA must know the key in order to encrypt the random value successfully.

b. This scheme does nothing to prove to the STA that the AP knows the key, so authentication is only one way.

c. If an attacker is eavesdropping, this scheme provides the attacker with a plaintext-ciphertext pair to use in cryptanalysis.

18.3 a.

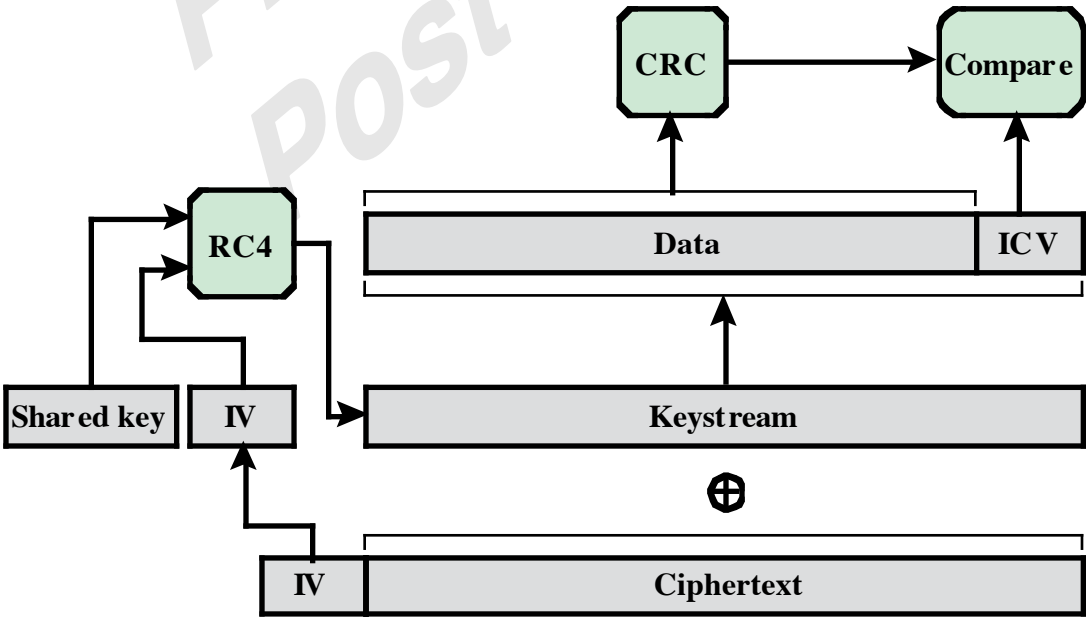


b. 1. The IV value, which is received in plaintext, is concatenated with the WEP key shared by transmitter and receiver to form the seed, or key input, to RC4.

2. The ciphertext portion of the received MPDU is decrypted using RC4 to recover the Data block and the ICV.

3. The ICV is computed over the plaintext received Data block and compared to the received plaintext ICV to authenticate the Data block.

c.



18.4 Because WEP works by XORing the data to get the ciphertext, bit flipping survives the encryption process. Flipping a bit in the plaintext always flips the same bit in the ciphertext and vice versa.

CHAPTER 19 ELECTRONIC MAIL SECURITY

ANSWERS TO QUESTIONS

19.1 **RFC 5321** defines the fields in the outer SMTP envelope of an email message. **RFC 5322** defines the Internet Message Format, which is the format of the email message inside the SMTP envelope.

19.2 **SMTP** encapsulates an email message in an envelope and is used to relay the encapsulated messages from source to destination through multiple MTAs. **Multipurpose Internet Mail Extension (MIME)** is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP) or some other mail transfer protocol and RFC 5322 for electronic mail. MIME defines a number of content formats and transfer encodings.

19.3 **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
Content-Transfer-Encoding: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.

19.4 Base64 is an encoding scheme for binary data. Each group of three octets of binary data is mapped into four ASCII characters.

19.5 Many electronic mail systems only permit the use of blocks consisting of ASCII text.

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

19.7 Authentication, confidentiality, e-mail compatibility, and compression.

19.8 A detached signature is useful in several contexts. A user may wish to maintain a separate signature log of all messages sent or received. A detached signature of an executable program can detect subsequent

virus infection. Finally, detached signatures can be used when more than one party must sign a document, such as a legal contract. Each person's signature is independent and therefore is applied only to the document. Otherwise, signatures would have to be nested, with the second signer signing both the document and the first signature, and so on.

19.9 DomainKeys Identified Mail (DKIM) is a specification for cryptographically signing e-mail messages, permitting a signing domain to claim responsibility for a message in the mail stream.

ANSWERS TO PROBLEMS

19.1 If the mail data itself contains the character sequence "<CR><LF>.<CR><LF>", the SMTP-client will insert an additional period at the beginning of the line, thus transmitting it as "<CR><LF>..<CR><LF>".

19.2 Post Office Protocol (POP3) POP3 allows an email client (user agent) to download an email from an email server (MTA). POP3 user agents connect via TCP to the server (typically port 110). The user agent enters a username and password (either stored internally for convenience or entered each time by the user for stronger security). After authorization, the UA can issue POP3 commands to retrieve and delete mail.

As with POP3, Internet Mail Access Protocol (IMAP) also enables an email client to access mail on an email server. IMAP also uses TCP, with server TCP port 143. IMAP is more complex than POP3. IMAP provides stronger authentication than POP3 and provides other functions not supported by POP3.

19.3 The signature should be generated before compression for two reasons:

a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

b. Even if one were willing to generate dynamically a recompressed message for verification, the typical lossless compression algorithm presents a difficulty. The algorithm usually is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm

can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all S/MIME implementations to the same version of the compression algorithm.

19.4 Basically, the DNS can scale well with the growth of the Internet. A centralized database such as the HOSTS.TXT system would not be appropriate for today's Internet, as the file, and hence the costs of its distribution, would be too large. Because of its distributed nature, the DNS allows organizations to manage their own domain space, while the old HOSTS.TXT system required changes to be submitted to the maintainer of the centralized database, something unthinkable for a huge international network as today's Internet.

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

19.6 **a.** The first step is to convert the characters into 8-bit ASCII with zero parity. Consulting the table in Appendix Q, we have the following correspondence:

```
p    01110000
l    01101100
a    01100001
i    01101001
n    01101110
t    01110100
e    01100101
x    01111000
t    01110100
```

Next, we block these off into groups of 6 bits, show the 6-bit decimal value, and do the encoding.

```
011100 000110 110001 100001 011010 010110 111001 110100
  28      6     49     33     26     22     57     52
   c      G      x      h      a      W      5      0
011001 010111 100001 110100
  25     23     33     52
   Z      X      h      0
```

So the radix-64 encoding is cGxhaW50ZXh0

b. All of the characters are "safe", so the quoted-printable encoding is simply plaintext

19.7

	Requires PKIX validation	Does not require PKIX validation
TLSA RR contains a trust a nchor that issued one of the certificates	PKIX-TA	DANE-TA
TLSA matches an end entity, or leaf certificate	PKIX-EE	DANE-EE

CHAPTER 20 IP SECURITY

ANSWERS TO QUESTIONS

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

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

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

A security association is normally defined by the following parameters:

Sequence Number Counter: A 32-bit value used to generate the Sequence Number field in AH or ESP headers, described in Section 20.3 (required for all implementations). **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number

Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations). **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay, described in Section 20.3 (required for all implementations). **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations). **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations). **Lifetime of this Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementations). **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations). These modes are discussed later in this section. **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

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

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

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

20.7 **Transport adjacency:** Refers to applying more than one security protocol to the same IP packet, without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one IPSec instance: the (ultimate) destination. **Iterated tunneling:** Refers to the application of multiple

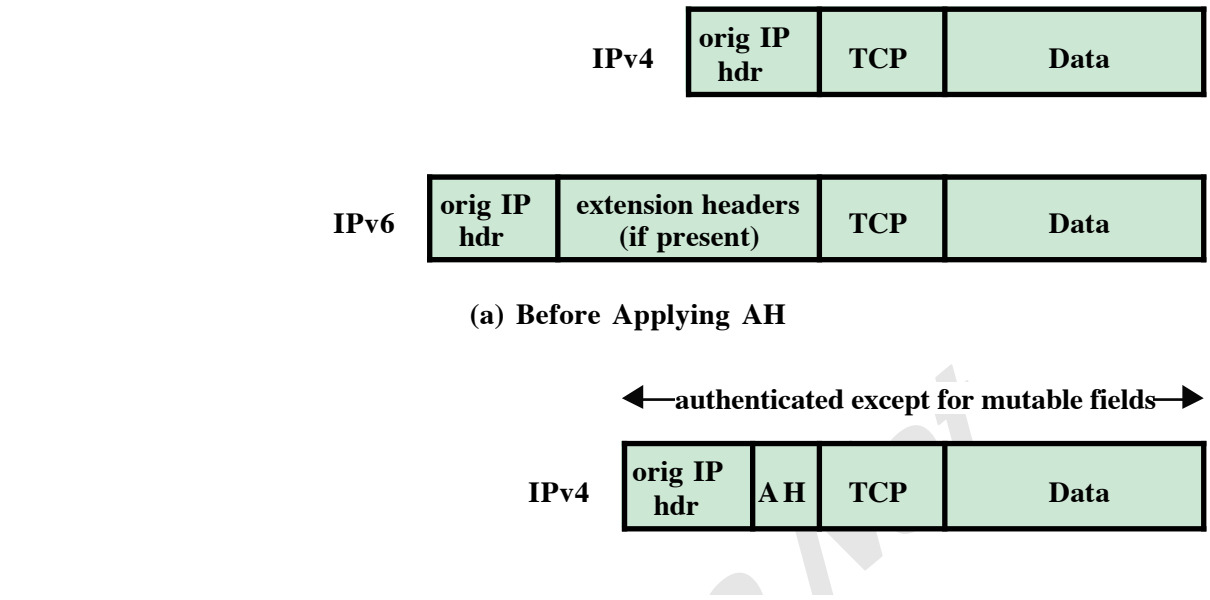
layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPSec site along the path.

20.8 **Oakley** is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats. **ISAKMP** provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

ANSWERS TO PROBLEMS

20.1
row 1: Traffic between this host and any other host, both using port 500, and using UDP, bypasses IPsec. This is used for IKE traffic.
row 2: ICMP message to or from any remote address are error messages, and bypass IPSec.
row 3: Traffic between 1.2.3.101 and 1.2.3.0/24 is intranet traffic and must be protected by ESP, with the exception of traffic defined in earlier rows.
row 4: TCP traffic between this host (1.2.3.101) and the server (1.2.4.10) on server port 80 is ESP protected.
row 5: TCP traffic between this host (1.2.3.101) and the server (1.2.4.10) on server port 80 is protected by TLS and so can bypass IPSec.
row 6: Any other traffic between 1.2.3.101 and 1.2.3.0/24 is prohibited and is discarded.
row 7: Any other traffic between 1.2.3.101 goes to the Internet and bypasses IPSec.

20.2.



20.3 **AH** provides access control, connectionless integrity, data origin authentication, and rejection of replayed packets. **ESP** provides all of these plus confidentiality and limited traffic flow confidentiality.

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

Mutable but predictable: Destination Address (with loose or strict source routing). At each intermediate router designated in the source routing list, the Destination Address field is changed to indicate the next designated address. However, the source routing field contains the information needed for doing the MAC calculation.
Mutable (zeroed prior to ICV calculation): Type of Service (TOS), Flags, Fragment Offset, Time to Live (TTL), Header Checksum. TOS may be altered by a router to reflect a reduced service. Flags and Fragment offset are altered if an router performs fragmentation. TTL is decreased at each router. The Header Checksum changes if any of these other fields change.

b. **Immutable**: Version, Payload Length, Next Header (This should be the value for AH.), Source Address, Destination Address (without Routing Extension Header)
Mutable but predictable: Destination Address (with Routing Extension Header)
Mutable (zeroed prior to ICV calculation): Class, Flow Label, Hop Limit

c. IPv6 options in the Hop-by-Hop and Destination Extension Headers contain a bit that indicates whether the option might change (unpredictably) during transit.
Mutable but predictable: Routing
Not Applicable: Fragmentation occurs after outbound IPSec processing and reassembly occur before inbound IPSec processing , so the Fragmentation Extension Header, if it exists, is not seen by IPSec.

20.5 **a.** The received packet is to the left of the window, so the packet is discarded; this is an auditable event. No change is made to window parameters.

b. The received packet falls within the window. If it is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked. If it is not new, the packet is discarded. In either case, no change is made to window parameters.

c. The received packet is to the right of the window and is new, so the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked. In this case, the window now spans from 120 to 540.

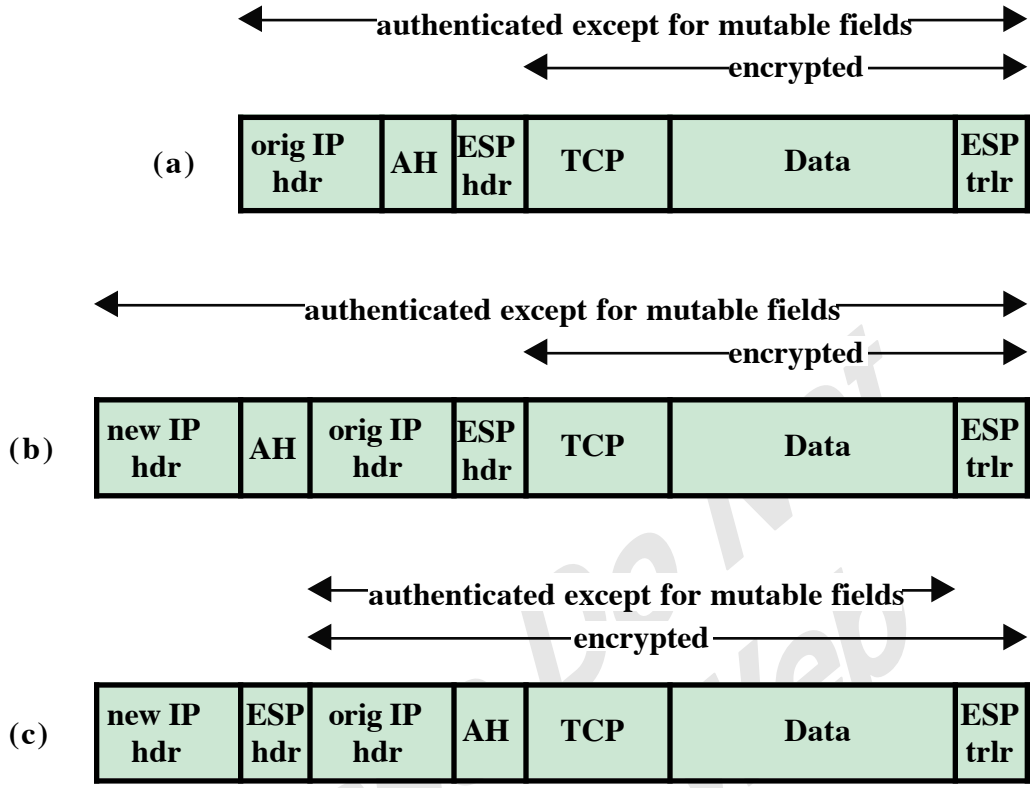
20.6 From RFC 2401

IPv4 Header Fields	Outer Header at Encapsulator	Inner Header at Decapsulator
version	4 (1)	no change
header length	constructed	no change
TOS	copied from inner header (5)	no change
total length	constructed	no change
ID	constructed	no change
Flags	constructed, DF (4)	no change
Fragment offset	constructed	no change
TTL	constructed	decrement (2)
protocol	AH, ESP, routing header	no change
checksum	constructed	no change
source address	constructed (3)	no change
destination address	constructed (3)	no change
options	never copied	no change

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

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

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



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

20.9 The Initial Exchanges and the CREATE_CHILD_SA Exchange

20.10 It is an addition to the IP layer.

CHAPTER 21 NETWORK ENDPOINT SECURITY

ANSWERS TO QUESTIONS

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

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

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

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

21.5 A **traditional packet filter** makes filtering decisions on an individual packet basis and does not take into consideration any higher layer context. A **stateful inspection packet filter** tightens up the rules for TCP traffic by creating a directory of outbound TCP connections, as shown in Table 23.2. There is an entry for each currently established connection. The packet filter will now allow incoming traffic to high-numbered ports only for those packets that fit the profile of one of the entries in this directory

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

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

21.8 Between internal and external firewalls are one or more networked devices in a region referred to as a DMZ (demilitarized zone) network. Systems that are externally accessible but need some protections are usually located on DMZ networks. Typically, the systems in the DMZ require or foster external connectivity, such as a corporate Web site, an e-mail server, or a DNS (domain name system) server.

21.9 An **external firewall** is placed at the edge of a local or enterprise network, just inside the boundary router that connects to the Internet or some wide area network (WAN). One or more **internal firewalls** protect the bulk of the enterprise network.

21.10 **Host-based IDS:** Monitors the characteristics of a single host and the events occurring within that host for suspicious activity
Network-based IDS: Monitors network traffic for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity

21.11 **Sensors:** Sensors are responsible for collecting data. The input for a sensor may be any part of a system that could contain evidence of an intrusion. Types of input to a sensor include network packets, log files, and system call traces. Sensors collect and forward this information to the analyzer.
Analyzers: Analyzers receive input from one or more sensors or from other analyzers. The analyzer is responsible for determining if an intrusion has occurred. The output of this component is an indication that an intrusion has occurred. The output may include evidence supporting the conclusion that an intrusion occurred. The analyzer may provide guidance about what actions to take as a result of the intrusion.
User interface: The user interface to an IDS enables a user to view output from the system or control the behavior of the system. In some systems, the user interface may equate to a manager, director, or console component.

21.12 **Misuse detection** is based on rules that specify system events, sequences of events, or observable properties of a system that are believed to be symptomatic of security incidents. Misuse detectors use various pattern-matching algorithms, operating on large databases of attack patterns, or *signatures*.
Anomaly detection searches for activity that is different from the normal behavior of system entities and system resources.

21.13 **Virus**: A computer program that can copy itself and infect a computer without permission or knowledge of the user.

Worm: A self-replicating, self-propagating, self-contained program that uses networking mechanisms to spread itself.
Trojan Horse: A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the program.
Spyware: Software that is secretly or surreptitiously installed into an information system to gather information on individuals or organizations without their knowledge
Rootkit: A set of tools used by an attacker after gaining root-level access to a host to conceal the attacker's activities on the host and permit the attacker to maintain root-level access to the host through covert means.
Backdoor: An undocumented way of gaining access to a computer system. Typically, a backdoor is a program that has the ability to bypass a system's security control, allowing an attacker to access the system stealthily.
Mobile code: Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Bot: Also known as a zombie.

21.14 **Network traffic analysis** involves monitoring traffic flows to detect potentially malicious activity.
Payload analysis is a real-time or near-real-time activity that involves looking for known malicious payloads (signature detection) or for looking payload patterns that are anomalous.
Endpoint analysis involves a wide variety of tools and approaches implemented at the endpoint.

21.15 DDoS attacks make computer systems inaccessible by flooding servers, networks, or even end-user systems with useless traffic so that legitimate users can no longer gain access to those resources.

ANSWERS TO PROBLEMS

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

21.2 When a TCP packet is fragmented so as to force interesting header fields out of the zero-offset fragment, there must exist a fragment with FO equal to 1. If a packet with FO = 1 is seen, conversely, it could indicate the presence, in the fragment set, of a zero-offset fragment with a transport header length of eight octets Discarding this one-

offset fragment will block reassembly at the receiving host and be as effective as the direct method described above.

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

21.4
1. Allow return TCP Connections to internal subnet.
2. Prevent Firewall system itself from directly connecting to anything.
3. Prevent External users from directly accessing the Firewall system.
4. Internal Users can access External servers,
5. Allow External Users to send email in.
6. Allow External Users to access WWW server.
7. Everything not previously allowed is explicitly denied.

21.5
a. Rules A and B allow inbound SMTP connections (incoming email) Rules C and D allow outbound SMTP connections (outgoing email) Rule E is the default rule that applies if the other rules do not apply.
b. Packet 1: Permit (A); Packet 2: Permit (B): Packet 3: Permit (C) Packet 4: Permit (D)
c. The attack could succeed because in the original filter set, rules B and D allow all connections where both ends are using ports above 1021.

21.6
a. A source port is added to the rule set.
b. Packet 1: Permit (A); Packet 2: Permit (B): Packet 3: Permit (C) Packet 4: Permit (D); Packet 5: Deny (E); Packet 6: Deny (E)

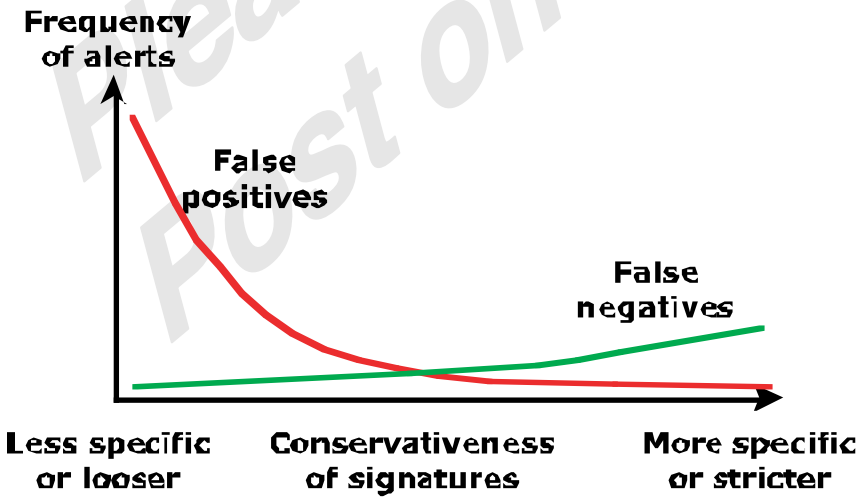
21.7
a. Packet 7 is admitted under rule D. Packet 8 is admitted under rule C.
b. Add a column called ACK Set, with the following values for each rule: A = Yes; B = Yes; C = Any; D = Yes; E = Any

21.8 A requirement like "all external Web traffic must flow via the organization's Web proxy." is easier stated than implemented. This is because identifying what actually constitutes “web traffic” is highly problematical. Although the standard port for HTTP web servers is port 80, servers are found on a large number of other ports (including servers belonging to large, well-known and widely used organizations). This means it is very difficult to block direct access to all possible web servers just using port filters. Whilst it is easy enough to configure web browser programs to always use a proxy, this will not stop direct access by other programs. It also means that the proxy server must have access to a very large number of external ports, since otherwise access to some servers would be limited. As well as HTTP access, other protocols are used on the web. All of these should also be directed via

the proxy in order to implement the desired policy. But this may impact the operation of other programs using these protocols. In particular, the HTTPS protocol is used for secure web access that encrypts all traffic flowing between the client and the server. Since the traffic is encrypted, it means the proxy cannot inspect its contents in order to apply malware, SPAM or other desired filtering. Whilst there are some mechanisms for terminating the encrypted connections at the proxy, they have limitations and require the use of suitable browsers and proxy servers.

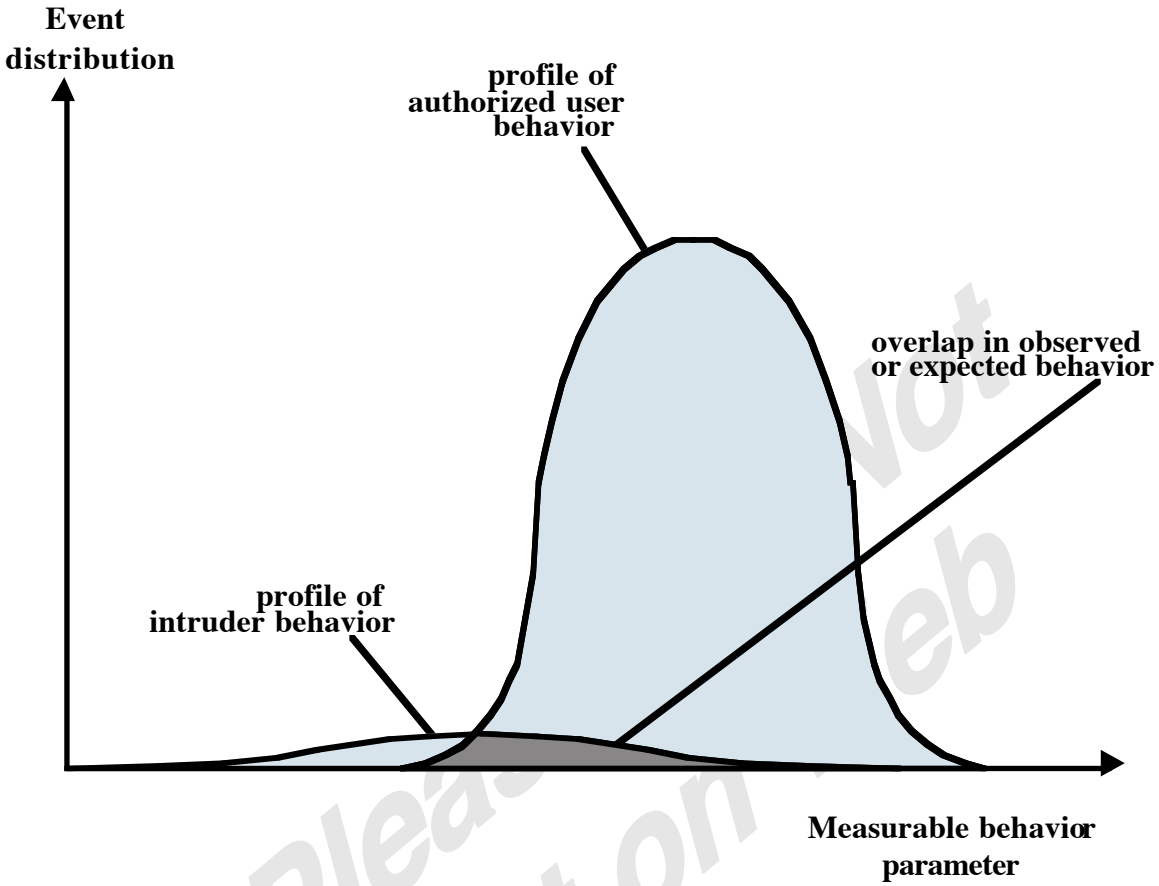
21.9 A possible requirement to manage information leakage requires all external e-mail to be given a sensitivity tag (or classification) in its subject and for external e-mail to have the lowest sensitivity tag. At its simplest a policy can just require user's to always include such a tag in email messages. Alternatively with suitable email agent programs it may be possible to enforce the prompting for and inclusion of such a tag on message creation. Then, when external email is being relayed through the firewall, the mail relay server must check that the correct tag value is present in the Subject header, and refuse to forward the email outside the organization if not, and notify the user of its rejection.

21.10 This is a typical example:



22.11 **a.** The graph below doesn't look like a correct probability distribution and is instead labeled as *event distribution*. The point here is that even if you have nice, mostly non-overlapping probability distributions for distinguishing intruders and authorized users like Figure 22.1, the problem is for most systems we hope the actual numbers of intruders is dwarfed by the number of authorized users. This means that the long tail of the authorized user's distribution that overlaps with the

intruder's distribution would generate lots of false positives (relative to the number of real intruders detected) even if it is only a few percent of the authorized users.



b. A randomly selected event that in the overlap region is (roughly) 95% likely to be an authorized user, even though the region covers 50% of the intruder's probability distribution.

22.12 A file integrity checking tool such as tripwire can be very useful in identifying changed files or directories on a system, particularly when those change should not have occurred. However most computer systems are not static, and significant numbers of files do change constantly. Hence it is necessary to configure tripwire with a list of files and directories to monitor, since otherwise reports to the administrator would be filled with lists of files that are changing as a matter of normal operation of the system. It is not too difficult to monitor a small list of critical system programs, daemons and configuration files. Doing this means attempts to alter these files will likely be detected. However the large areas of the system not being monitored means an attacker changing or adding files in these areas will not be detected. The more of the system that is to be monitored, the more care is needed to identify only files not expected to change. Even then, it is likely that user's home areas, and other shared document areas, cannot be monitored, since they are likely to be

creating and changing files in there regularly. As well, there needs to be a process to manage the update of monitored files (as a result of installing patches, upgrades, new services, configuration changes etc). This process has to verify that the changed files are correct, and then update the cryptographic checksums of these files. Lastly the database of cryptographic checksums must be protected from any attempt by an attacker to corrupt it, ideally by locating on read-only media (except when controlled updates are occurring).

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

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

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

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

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

CHAPTER 22 CLOUD SECURITY

ANSWERS TO QUESTIONS

22.1 A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

22.2 **Software as a service (SaaS)**: Provides service to customers in the form of software, specifically application software, running on and accessible in the cloud.
Platform as a service (PaaS): Provides service to customers in the form of a platform on which the customer's applications can run.
Infrastructure as a service (IaaS): Provides the customer access to the underlying cloud infrastructure.

22.3 The NIST cloud computing reference architecture focuses on the requirements of "what" cloud services provide, not a "how to" design solution and implementation. The reference architecture is intended to facilitate the understanding of the operational intricacies in cloud computing. It does not represent the system architecture of a specific cloud computing system; instead it is a tool for describing, discussing, and developing a system-specific architecture using a common framework of reference.

22.4 **Abuse and nefarious use of cloud computing**: For many CPs, it is relatively easy to register and begin using cloud services, some even offering free limited trial periods. This enables attackers to get inside the cloud to conduct various attacks, such as spamming, malicious code attacks, and denial of service.
Insecure interfaces and APIs: CPs expose a set of software interfaces or APIs that customers use to manage and interact with cloud services. The security and availability of general cloud services is dependent upon the security of these basic APIs. From authentication and access control to encryption and activity monitoring, these interfaces must be designed to protect against both accidental and malicious attempts to circumvent policy.
Malicious insiders: Under the cloud computing paradigm, an organization relinquishes direct control over many aspects of security and, in doing so, confers an unprecedented level of trust onto the CP.

One grave concern is the risk of malicious insider activity. Cloud architectures necessitate certain roles that are extremely high-risk.

Shared technology issues: IaaS vendors deliver their services in a scalable way by sharing infrastructure. Often, the underlying components that make up this infrastructure (CPU caches, GPUs, etc.) were not designed to offer strong isolation properties for a multi-tenant architecture. CPs typically approach this risk by the use of isolated virtual machines for individual clients. This approach is still vulnerable to attack, by both insiders and outsiders, and so can only be a part of an overall security strategy.

Data loss or leakage: For many clients, the most devastating impact from a security breach is the loss or leakage of data. We address this issue in the next section.

Account or service hijacking: Account and service hijacking, usually with stolen credentials, remains a top threat. With stolen credentials, attackers can often access critical areas of deployed cloud computing services, allowing them to compromise the confidentiality, integrity, and availability of those services.

Unknown risk profile: In using cloud infrastructures, the client necessarily cedes control to the cloud provider on a number of issues that may affect security. Thus the client must pay attention to and clearly define the roles and responsibilities involved for managing risks. For example, employees may deploy applications and data resources at the CP without observing the normal policies and procedures for privacy, security, and oversight.

22.5 OpenStack is an open source software project of the OpenStack Foundation that aims to produce an open source cloud operating system.

CHAPTER 33 IOT SECURITY

ANSWERS TO QUESTIONS

23.1 The Internet of things (IoT) is a term that refers to the expanding interconnection of smart devices, ranging from appliances to tiny sensors

23.2 **Sensor**: A sensor measures some parameter of a physical, chemical, or biological entity and delivers an electronic signal proportional to the observed characteristic, either in the form of an analog voltage level or a digital signal. In both cases, the sensor output is typically input to a microcontroller or other management element.
Actuator: An actuator receives an electronic signal from a controller and responds by interacting with its environment to produce an effect on some parameter of a physical, chemical, or biological entity.
Microcontroller: The "smart" in a smart device is provided by a deeply embedded microcontroller.
Transceiver: A transceiver contains the electronics needed to transmit and receive data. Most IoT devices contain a wireless transceiver, capable of communication using Wi-Fi, ZigBee, or some other wireless scheme.
Radio-Frequency Identification (RFID): (RFID) technology, which uses radio waves to identify items, is increasingly becoming an enabling technology for IoT. The main elements of an RFID system are tags and readers. RFID tags are small programmable devices used for object, animal, and human tracking. They come in a variety of shapes, sizes, functionalities, and costs. RFID readers acquire and sometimes rewrite information stored on RFID tags that come within operating range (a few inches up to several feet). Readers are usually connected to a computer system that records and formats the acquired information for further uses.

23.3 The billions of IoT devices have various security vulnerabilities and there is no effective way to patch these in a timely manner.

23.4 **Tamper resistant**: A characteristic of a system component that provides passive protection against an attack.
Tamper detection: Techniques to ensure that the overall system is made aware of unwanted physical access

23.5 MiniSec is an open-source security module that is part of the TinyOS operating system.