# CHAPTER 10

## PRACTICE SET

### Questions

**Q10-1.** Only snooping is a threat to confidentiality. Masquerading and repudiation are threats to integrity.

**Q10-3.** Only denial of service is a threat to availability. Repudiation and modification are threats to integrity.

**Q10-5.** This is an example of steganography because the letter is covered by the seal.

**Q10-7.** This is an example of steganography because the message is hidden inside the essay.

**Q10-9.** This is an example of symmetric-key cryptography because the same key is used both for encryption and decryption. In asymmetric-key cryptography, a public key should be used for encryption and a private key for decryption.

**Q10-11.** This is a monoalphabetic substitution cipher because changing depends on what the character is, not on the position of the character.

**Q10-13.** The key in this case needs to be between 0 and 25 (although there is no cipher if the key is 0). on average Eve needs to test 26/2 = 13 keys to break the code.

**Q10-15.** For the additive cipher, we use only one key. For the monoalphabetic cipher, we use 26 keys. For the autokey cipher, we use 1000 keys.

**Q10-17.** This is a straight permutation because the number of inputs and the number of outputs are the same.

**Q10-19.**

- **a.** The inverse of a swap operation is another swap operation. In other words, the swap operation is self-invertible.
- **b.** The inverse of a shift right operation is a shift left operation using the same number of bits to be shifted.
- **c.** The inverse of a combine operation is a split operation.

**Q10-21.** Half of the input in DES is only 32 bits. The key is applied only to this 32-bit section. The two inputs to an XOR should be of the same size. Since the key is 48 bits, we need to expand the 32-bit section to a 48-bit section using an expansion P-box. Making the key for each round smaller jeopardizes the security of DES.

**Q10-23.** The one-time pad is an ideal, but a theoretical, cipher; it cannot be implemented easily because it requires that the sender and receiver have a secure channel to exchange the key stream. However, it is controversial; if there is a secure channel, the main message, in plaintext, can be sent through that channel.

**Q10-25.** To be safe, the keys used in asymmetric-key cryptography should be very large. This makes the calculation very long if the message is also long. Compare the short key of DES (56 bits) with the long key of RSA (500 to 1000 bits).

**Q10-27.** If $e = 1$, there is no encryption: $C = P^1 = P$. If Eve intercepts the ciphertext, she has the plaintext.

**Q10-29.** Message authentication and entity authentication both authenticate the sender for the receiver. As their names indicate, however, message authentication authenticates the sender for a particular message, while entity authentication authenticates the sender for the entire session in which several messages can be sent. If the sender is sending only a single message, message authentication has the same effect as entity authentication; if a sender is sending several messages, one entity authentication authenticates the sender for all messages.

**Q10-31.** The first choice is actually very inefficient and not practical because Alice needs to have a shared secret key between herself and each of the recipients. She also needs to create fifty separate MACs, one for each secret key. Alice can use digital signature and sign only one message with her own private key. She then sends a copy of the message to each recipient. Each recipient needs to use Alice's public key to verify the message.

**Q10-33.** Alice needs 101 keys. She needs her own private key to sign the document. She needs 100 public keys of the recipients to encrypt the signed document.

**Q10-35.** A key distribution center (KDC) is designed to solve the problem of using secret keys. The center creates a secret key between the center and each member. If two members need to communicate with each other, the center establishes a session secret key between the two. The session key is temporary and used only once.

## Problems

**P10-1.**

**a.** This is **snooping** (an attack to confidentiality). Although the contents of the test are not confidential on the day of the test, they are confidential before the test day.

**b.** This is **modification** (an attack to integrity). The value of the check is changed (from $10 to $100).

**c.** This is **denial of service** (an attack to availability). Sending so many e-mails may crash the server and the service may be interrupted.

**P10-3.**

**a.** The ciphertext is **NBCMCMUHYRYLWCMY** as shown below:

| Plaintext | | | Encryption | Ciphertext | | |
|---|---|---|---|---|---|---|
| t | $\rightarrow$ | 19 | (19 + 20) mod 26 = 13 | 13 | $\rightarrow$ | N |
| h | $\rightarrow$ | 07 | (07 + 20) mod 26 = 01 | 01 | $\rightarrow$ | B |
| i | $\rightarrow$ | 08 | (08 + 20) mod 26 = 02 | 02 | $\rightarrow$ | C |
| s | $\rightarrow$ | 18 | (18 + 20) mod 26 = 12 | 12 | $\rightarrow$ | M |
| i | $\rightarrow$ | 08 | (08 + 20) mod 26 = 02 | 02 | $\rightarrow$ | C |
| s | $\rightarrow$ | 18 | (18 + 20) mod 26 = 12 | 12 | $\rightarrow$ | M |
| a | $\rightarrow$ | 00 | (00 + 20) mod 26 = 20 | 20 | $\rightarrow$ | U |
| n | $\rightarrow$ | 13 | (13 + 20) mod 26 = 07 | 07 | $\rightarrow$ | H |
| e | $\rightarrow$ | 04 | (04 + 20) mod 26 = 24 | 24 | $\rightarrow$ | Y |
| x | $\rightarrow$ | 23 | (23 + 20) mod 26 = 17 | 17 | $\rightarrow$ | R |
| e | $\rightarrow$ | 04 | (04 + 20) mod 26 = 24 | 24 | $\rightarrow$ | Y |
| r | $\rightarrow$ | 17 | (17 + 20) mod 26 = 11 | 11 | $\rightarrow$ | L |
| c | $\rightarrow$ | 02 | (02 + 20) mod 26 = 22 | 22 | $\rightarrow$ | W |
| i | $\rightarrow$ | 08 | (08 + 20) mod 26 = 02 | 02 | $\rightarrow$ | C |
| s | $\rightarrow$ | 18 | (18 + 20) mod 26 = 12 | 12 | $\rightarrow$ | M |
| e | $\rightarrow$ | 04 | (04 + 20) mod 26 = 24 | 24 | $\rightarrow$ | Y |

**b.** We can retrieve the plaintext by subtracting 20 from each ciphertext character using modulo 26 arithmetic as shown below:

| Ciphertext | | | Decryption | Plaintext | | |
|---|---|---|---|---|---|---|
| N | $\rightarrow$ | 13 | (13 – 20) mod 26 = 19 | 19 | $\rightarrow$ | t |
| B | $\rightarrow$ | 01 | (01 – 20) mod 26 = 07 | 07 | $\rightarrow$ | h |
| C | $\rightarrow$ | 02 | (02 – 20) mod 26 = 08 | 08 | $\rightarrow$ | i |
| M | $\rightarrow$ | 12 | (12 – 20) mod 26 = 18 | 18 | $\rightarrow$ | s |
| C | $\rightarrow$ | 02 | (02 – 20) mod 26 = 08 | 08 | $\rightarrow$ | i |
| M | $\rightarrow$ | 12 | (12 – 20) mod 26 = 18 | 18 | $\rightarrow$ | s |
| U | $\rightarrow$ | 20 | (20 – 20) mod 26 = 00 | 00 | $\rightarrow$ | a |
| H | $\rightarrow$ | 07 | (07 – 20) mod 26 = 13 | 13 | $\rightarrow$ | n |
| Y | $\rightarrow$ | 24 | (24 – 20) mod 26 = 04 | 04 | $\rightarrow$ | e |
| R | $\rightarrow$ | 17 | (17 – 20) mod 26 = 23 | 23 | $\rightarrow$ | x |
| Y | $\rightarrow$ | 24 | (24 – 20) mod 26 = 04 | 04 | $\rightarrow$ | e |
| L | $\rightarrow$ | 11 | (11 – 20) mod 26 = 17 | 17 | $\rightarrow$ | r |
| W | $\rightarrow$ | 22 | (22 – 20) mod 26 = 02 | 02 | $\rightarrow$ | c |
| C | $\rightarrow$ | 02 | (02 – 20) mod 26 = 08 | 08 | $\rightarrow$ | i |
| M | $\rightarrow$ | 12 | (12 – 20) mod 26 = 18 | 18 | $\rightarrow$ | s |
| Y | $\rightarrow$ | 24 | (24 – 20) mod 26 = 04 | 04 | $\rightarrow$ | e |

**P10-5.** The plaintext and ciphertext are shown below, ignoring the space:

| Plaintext | Ciphertext |
|---|---|
| an exercise | (5, 5), (3, 3), (1, 5), (3, 1), (1, 5), (2, 2), (3, 5), (4, 4), (3, 2), (1, 5) |

**P10-7.** We apply the key = 1, 2, 3, 4, 5, 6, and 7 to find the plaintext.

| Ciphertext | Key | Plaintext |
|---|---|---|
| UVACLYZLJBL | 1 | tuzbkxeykiaxy |
| UVACLYZLJBL | 2 | styajwdxjhzwj |
| UVACLYZLJBL | 3 | rsxzivcwigyvi |
| UVACLYZLJBL | 4 | qrwyhubvhfxuh |
| UVACLYZLJBL | 5 | pqvxgtaugewtg |
| UVACLYZLJBL | 6 | opuwfsztfdvsf |
| UVACLYZLJBL | 7 | notverysecure |

**P10-9.** The content of each element represents the input column number; the index represents the output column number.

**a.** The encryption key can be represented as shown below.

|  |  |  |  |  |
|---|---|---|---|---|
| 3 | 1 | 4 | 5 | 2 |
| 1 | 2 | 3 | 4 | 5 |

**b.** The decryption key can be represented as shown below. Note that the indexes are sorted.

|  |  |  |  |  |
|---|---|---|---|---|
| 2 | 5 | 1 | 3 | 4 |
| 1 | 2 | 3 | 4 | 5 |

**c.** We can first exchange the content and index and then sort the array according to the index. For example, to get the decryption key from the encryption key in part a, we can first swap the contents and the index and then sort it according to the index.

**1: Swap**

|  |  |  |  |  |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 3 | 1 | 4 | 5 | 2 |

**2: Sort**

|  |  |  |  |  |
|---|---|---|---|---|
| 2 | 5 | 1 | 3 | 4 |
| 1 | 2 | 3 | 4 | 5 |

**P10-11.**

**a.** Swap of (**1001**1011) is (1011**1001**).

**b.** Swap of (1011**1001**) is (**1001**1011).

**c.** The original word in part *a* and the result of part *b* are the same, which shows that swapping is a self-invertible operation.

**P10-13.** As we said about the keys for the transposition cipher, permutation boxes (P-boxes) can be represented by a table in which the contents of each element shows the input number and the index show the output number.

**a.** A straight permutation box of $n \times n$ size is a table of $n$ entries in which each entry is unique. A compression permutation of $n \times m$ size is a table of $m$ entries in which the blocked inputs are not shown. An expansion permutation of $n \times m$ size is a table of $n$ entries in which some of the entries are repeated. The following shows the three tables for permutation boxes in Figure 10.8 in the text. We have not shown the indexes.

| | | | | |
|---|---|---|---|---|
| 2 | 5 | 4 | 1 | 3 |
| 1 | 2 | 3 | 4 | 5 |

**Straight**

| | | |
|---|---|---|
| 1 | 3 | 5 |
| 1 | 2 | 3 |

**Compression**

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 3 | 1 | 2 |
| 1 | 2 | 3 | 4 | 5 |

**Expansion**

**b.** The compression and expansion boxes have no inversions because the number of inputs and outputs is not equal. Only the straight boxes can be inverted. We first need to swap the contents and the indexes and then sort the result according to the index. The following shows the inversion of a straight permutation box used in the decryption.

**1: Original**

| | | | | |
|---|---|---|---|---|
| 2 | 5 | 4 | 1 | 3 |
| 1 | 2 | 3 | 4 | 5 |

→

**2: Swap**

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 2 | 5 | 4 | 1 | 3 |

→

**3: Reorder**

| | | | | |
|---|---|---|---|---|
| 4 | 1 | 5 | 3 | 2 |
| 1 | 2 | 3 | 4 | 5 |

**P10-15.** We need to show that P2 = P1 although NI is not invertible. In other words, we need to show that the component NI is eliminated in the decryption process.

**Encryption:** $\mathbf{C1 = P1 \oplus NI}$

**Decryption:** $\mathbf{P2 = C2 \oplus NI = C1 \oplus NI = (P1 \oplus NI) \oplus NI = P1 \oplus (NI \oplus NI)}$

$\mathbf{P2 = P1 \oplus (0) = P1}$

**P10-17.** Both operations are keyless and the operations are predefined. If Eve, the intruder, wants to break the cipher, she can easily simulate them.

**P10-19.** We first find the value of $\phi = (p - 1) \times (q - 1) = 120$. We then need to see which of the $e$ values satisfies the relation $(d \times e) \bmod 120 = 1$. We will find that 103 is the answer.

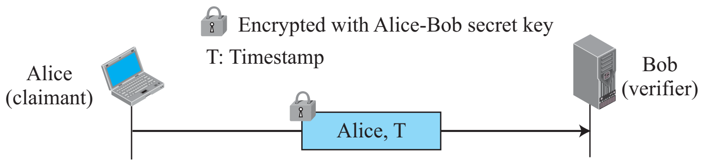
**a.** $(7 \times \mathbf{11}) \bmod 120 = 77$

**b.** $(7 \times \mathbf{103}) \bmod 120 = 721 \bmod 120 = \mathbf{1}$ **(Answer is 103)**

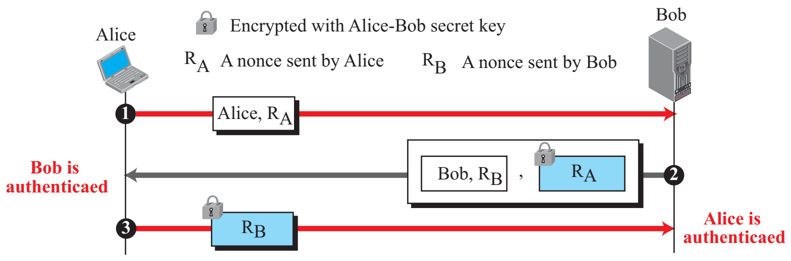
**c.** $(7 \times \mathbf{19}) \bmod 120 = 133 \bmod 120 = 13$

**P10-21.** Two messages can easily have the same traditional checksum. For example, two different messages in which two bytes are swapped will have the same traditional checksum because the calculation of a traditional checksum is independent from the position of the bytes. This means a traditional checksum cannot be used as a cryptographic hash function.

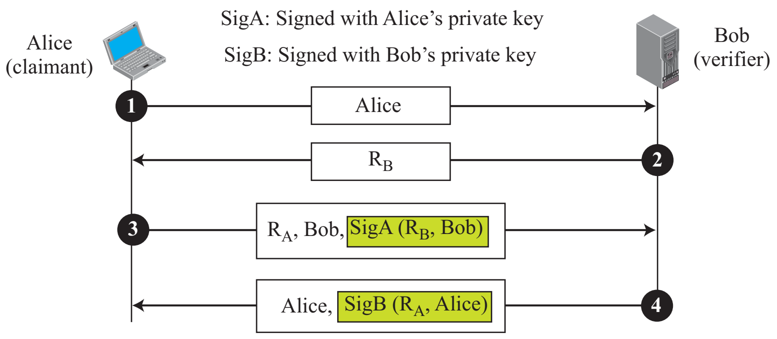
**P10-23.** The following shows the new diagram. The timestamp, T, included in the only message does the job of the challenge message and $R_B$.



**P10-25.** The following shows one simple solution. It shows the idea, but it is vulnerable to some attacks. There are some better but more complicated solutions. In the first message, Alice sends her identification and her nonce. In the second message, Bob sends his identification, his nonce, and Alice's encrypted nonce. Alice's nonce is encrypted with the shared secret key. When Alice receives this message and decrypts her nonce, Bob is authenticated for her because only Bob can encrypt Alice's nonce with the shared secret key. In the third message, Alice sends Bob's encrypted nonce. When Bob receives this message and decrypts his nonce, Alice is authenticated for Bob because only Alice can encrypt Bob's nonce with the shared secret key.


Encrypted with Alice-Bob secret key
$R_A$ A nonce sent by Alice
$R_B$ A nonce sent by Bob
Alice
Bob
Alice, $R_A$
Bob, $R_B$, $R_A$
$R_B$
Bob is authenticaed
Alice is authenticaed

**P10-27.** The following shows the new diagram. Two nonces and two keys are used to achieve bidirectional authentication.


SigA: Signed with Alice's private key
SigB: Signed with Bob's private key
Alice (claimant)
Bob (verifier)
Alice
$R_B$
$R_A$, Bob, SigA ($R_B$, Bob)
Alice, SigB ($R_A$, Alice)

**P10-29.** The following shows the value of the digest after hashing each character of the text. The method is not secure because the digest is always between 0 and 25. The total number of possible digests is N = 26.

| | |
|---|---|
| Initial value of the digest: | d = 00 |
| After hashing the first character (H: 07): | d = (00 + 07) mod 26 = 07 |
| After hashing the second character (E: 04): | d = (07 + 04) mod 26 = 11 |
| After hashing the third character (L: 11): | d = (11 + 11) mod 26 = 22 |
| After hashing the fourth character (L: 11): | d = (22 + 11) mod 26 = 07 |
| After hashing the fifth character (O: 14): | d = (07 + 14) mod 26 = 21 |

**P10-31.** Only applications that can create a session can use the service of SSL/TLS. There is no session in an e-mail. Alice sends an e-mail to Bob without creating a session.

**P10-33.** Alice creates the session key. She uses the session key to encrypt the message. She encrypts the session key with Bob's public key. Alice sends the encrypted message and the encrypted session key to Bob.

**P10-35.** Alice uses an *authenticatedData* object. She randomly creates a session key. She then encrypts the session key with Bob's public key. Alice now creates a MAC from the message hash and the secret key. Alice now sends the MAC, her public-key certificate (in case Bob needs to respond), the name of the algorithm used for creating the MAC, and the encrypted session key used to create the MAC. Alice also sends the message. The whole is referred to as the *authenticatedData*. (See Figure 10.38 in the book.)

**P10-37.** SSL provides both entity and message authentication. Two entities are authenticated for each other using the handshake protocol. The record protocol provides message authentication when it encapsulates messages from the application layer.

**P10-39.** Both parties that use PGP or S/MIME need to agree about the list of predefined cryptography algorithms. The sender of the e-mail defines the algorithms used for each purpose (confidentiality, integrity, and message authentication); the receiver needs to use those algorithms to be able to read the e-mail.

**P10-41.** An SA provides two services for IPSec: it creates a virtual connection and establishes security parameters between the two parties. The first service is not needed in the case of SSL because SSL runs over TCP, which is a connection-oriented protocol. The second service of SA is provided by the handshake protocol in SSL.

**P10-43.** The handshake protocol in SSL should start its function after the three-way handshaking in TCP because the handshaking protocol in SSL does not create a connection; it uses the connection established by TCP to exchange security parameters.

**P10-45.** IPSec provides both message authentication and entity authentication (see Table 10.1 in the text). At the beginning of each association, two parties are authenticated for each other based on their IP addresses. During data exchange, each packet is also authenticated.