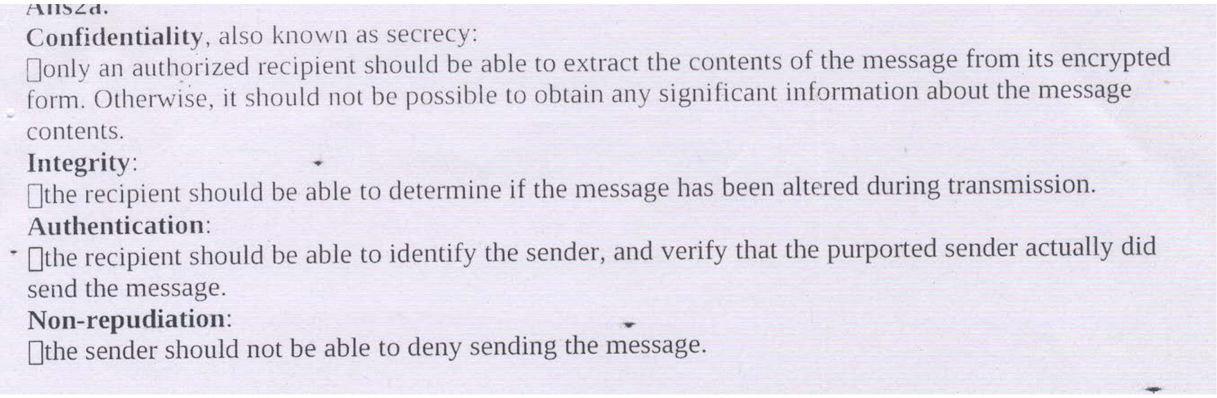
Q.2 a. Define and explain briefly four different services of security- Confidentiality, Integrity, Authentication and Non-repudiation.

Answer:



Ans2a.

Confidentiality, also known as secrecy:

□only an authorized recipient should be able to extract the contents of the message from its encrypted form. Otherwise, it should not be possible to obtain any significant information about the message contents.

Integrity:

□the recipient should be able to determine if the message has been altered during transmission.

Authentication:

□the recipient should be able to identify the sender, and verify that the purported sender actually did send the message.

Non-repudiation:

□the sender should not be able to deny sending the message.

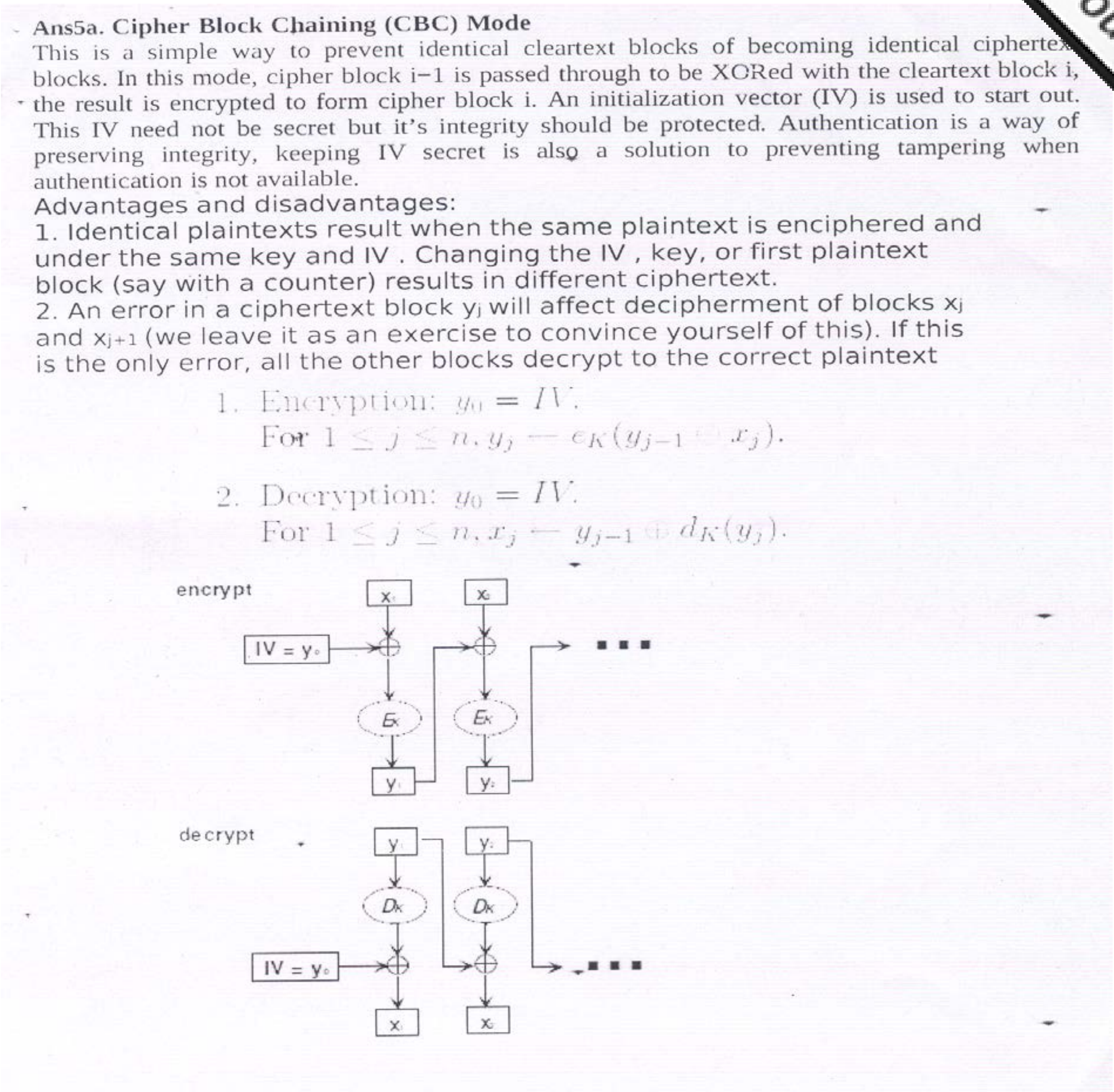
Q.3 c. Define a S-Box and mention the necessary condition for a S-Box to be invertible. What is the difference between Linear & Non-Linear S- Boxes?

Answer:

An S-box (Substitution box) can be thought of as a miniature substitution cipher. An S-box can be invertible if the number of input bits is same as the number of output bits.

Q.5 a. Explain CBC mode. Also list its advantages and disadvantages.

Answer:



Ans5a. Cipher Block Chaining (CBC) Mode

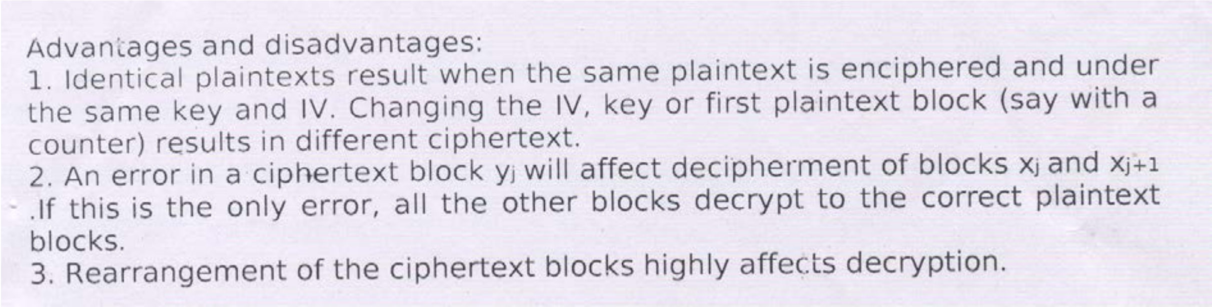
This is a simple way to prevent identical cleartext blocks of becoming identical ciphertex blocks. In this mode, cipher block i−1 is passed through to be XORed with the cleartext block i, the result is encrypted to form cipher block i. An initialization vector (IV) is used to start out. This IV need not be secret but it's integrity should be protected. Authentication is a way of preserving integrity, keeping IV secret is also a solution to preventing tampering when authentication is not available.

Advantages and disadvantages:

1. Identical plaintexts result when the same plaintext is enciphered and under the same key and IV . Changing the IV , key, or first plaintext block (say with a counter) results in different ciphertext.
2. An error in a ciphertext block y_j will affect decipherment of blocks x_j and x_{j+1} (we leave it as an exercise to convince yourself of this). If this is the only error, all the other blocks decrypt to the correct plaintext

1. Encryption: $y_0 = IV$.
   For $1 \leq j \leq n, y_j = e_K(y_{j-1} \oplus x_j)$.
2. Decryption: $y_0 = IV$.
   For $1 \leq j \leq n, x_j = y_{j-1} \oplus d_K(y_j)$.

encrypt
x₁
x₂
IV = y₀
Eₖ
Eₖ
y₁
y₂
decrypt
y₁
y₂
Dₖ
Dₖ
IV = y₀
x₁
x₂



Advantages and disadvantages:

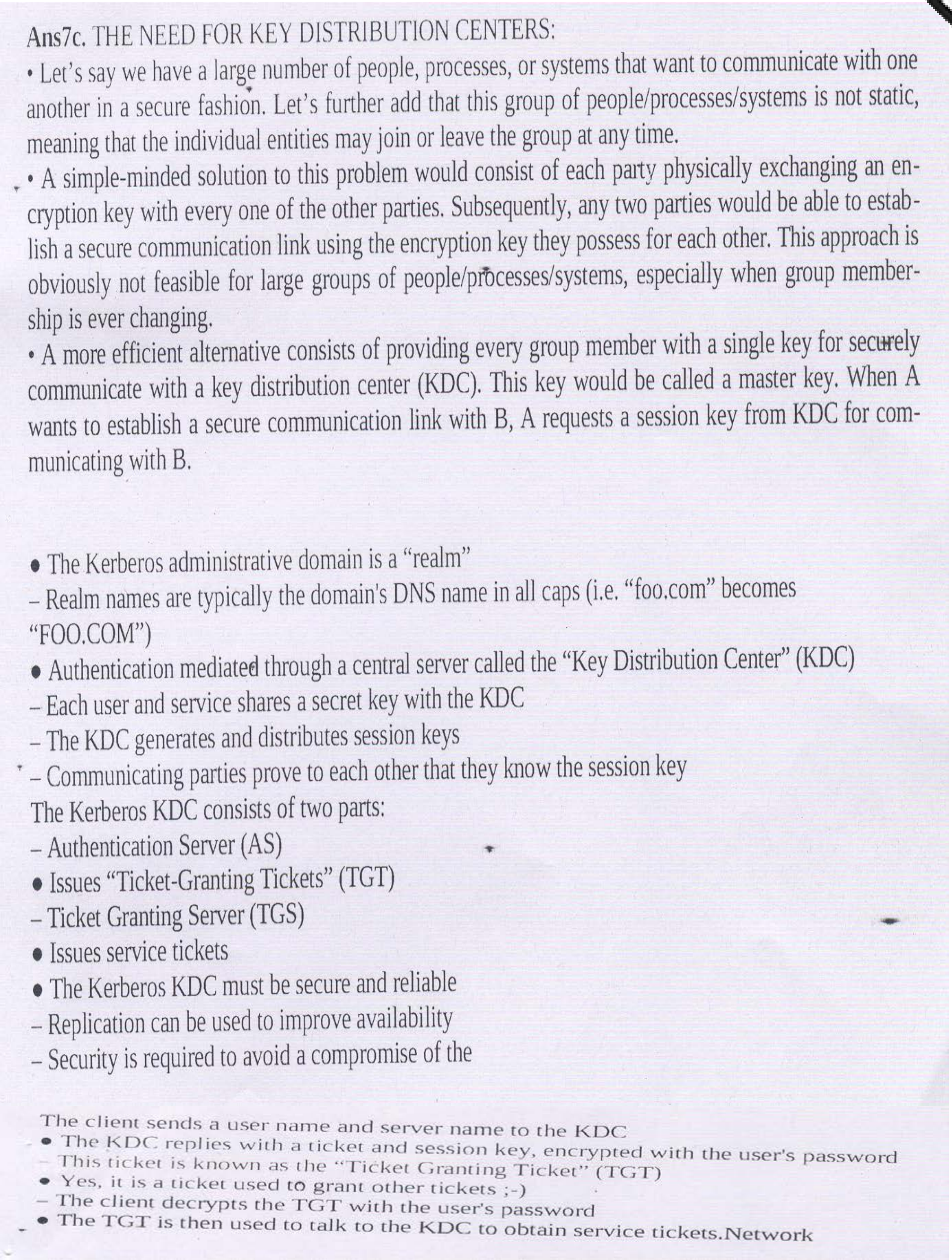
1. Identical plaintexts result when the same plaintext is enciphered and under the same key and IV. Changing the IV, key or first plaintext block (say with a counter) results in different ciphertext.
2. An error in a ciphertext block y_j will affect decipherment of blocks x_j and x_{j+1} .If this is the only error, all the other blocks decrypt to the correct plaintext blocks.
3. Rearrangement of the ciphertext blocks highly affects decryption.

Q.6 b. What is the minimum & maximum number of padding bits that can be added to a message? Explain.

Answer: Page Number 370 of Text Book

Q.7 c. What is the need for a key-distribution centre (KDC)?

Answer:



Ans7c. THE NEED FOR KEY DISTRIBUTION CENTERS:

• Let's say we have a large number of people, processes, or systems that want to communicate with one another in a secure fashion. Let's further add that this group of people/processes/systems is not static, meaning that the individual entities may join or leave the group at any time.

• A simple-minded solution to this problem would consist of each party physically exchanging an encryption key with every one of the other parties. Subsequently, any two parties would be able to establish a secure communication link using the encryption key they possess for each other. This approach is obviously not feasible for large groups of people/processes/systems, especially when group membership is ever changing.

• A more efficient alternative consists of providing every group member with a single key for securely communicate with a key distribution center (KDC). This key would be called a master key. When A wants to establish a secure communication link with B, A requests a session key from KDC for communicating with B.

● The Kerberos administrative domain is a "realm"

– Realm names are typically the domain's DNS name in all caps (i.e. "foo.com" becomes "FOO.COM")

● Authentication mediated through a central server called the "Key Distribution Center" (KDC)

– Each user and service shares a secret key with the KDC

– The KDC generates and distributes session keys

– Communicating parties prove to each other that they know the session key

The Kerberos KDC consists of two parts:

– Authentication Server (AS)

● Issues "Ticket-Granting Tickets" (TGT)

– Ticket Granting Server (TGS)

● Issues service tickets

● The Kerberos KDC must be secure and reliable

– Replication can be used to improve availability

– Security is required to avoid a compromise of the

The client sends a user name and server name to the KDC

● The KDC replies with a ticket and session key, encrypted with the user's password

– This ticket is known as the "Ticket Granting Ticket" (TGT)

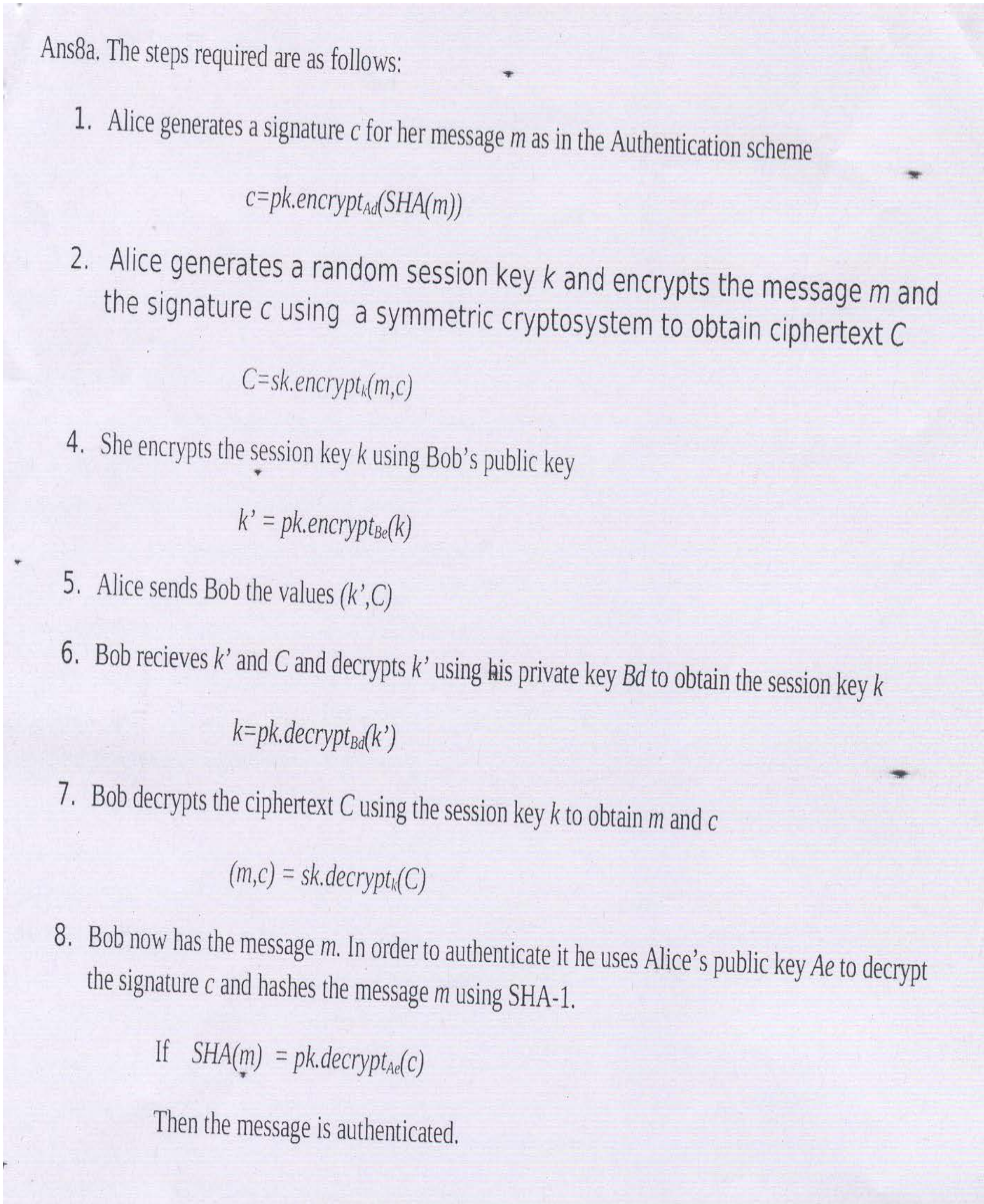
● Yes, it is a ticket used to grant other tickets ;-)

– The client decrypts the TGT with the user's password

● The TGT is then used to talk to the KDC to obtain service tickets.Network

Q.8 a. Explain how Bob and Alice exchange the secret key for encrypting messages in PGP.

Answer:



Ans8a. The steps required are as follows:

1. Alice generates a signature *c* for her message *m* as in the Authentication scheme

$$c=pk.encrypt_{Ad}(SHA(m))$$

2. Alice generates a random session key *k* and encrypts the message *m* and the signature *c* using a symmetric cryptosystem to obtain ciphertext *C*

$$C=sk.encrypt_{k}(m,c)$$

4. She encrypts the session key *k* using Bob's public key

$$k' = pk.encrypt_{Be}(k)$$

5. Alice sends Bob the values *(k',C)*

6. Bob recieves *k'* and *C* and decrypts *k'* using his private key *Bd* to obtain the session key *k*

$$k=pk.decrypt_{Bd}(k')$$

7. Bob decrypts the ciphertext *C* using the session key *k* to obtain *m* and *c*

$$(m,c) = sk.decrypt_{k}(C)$$

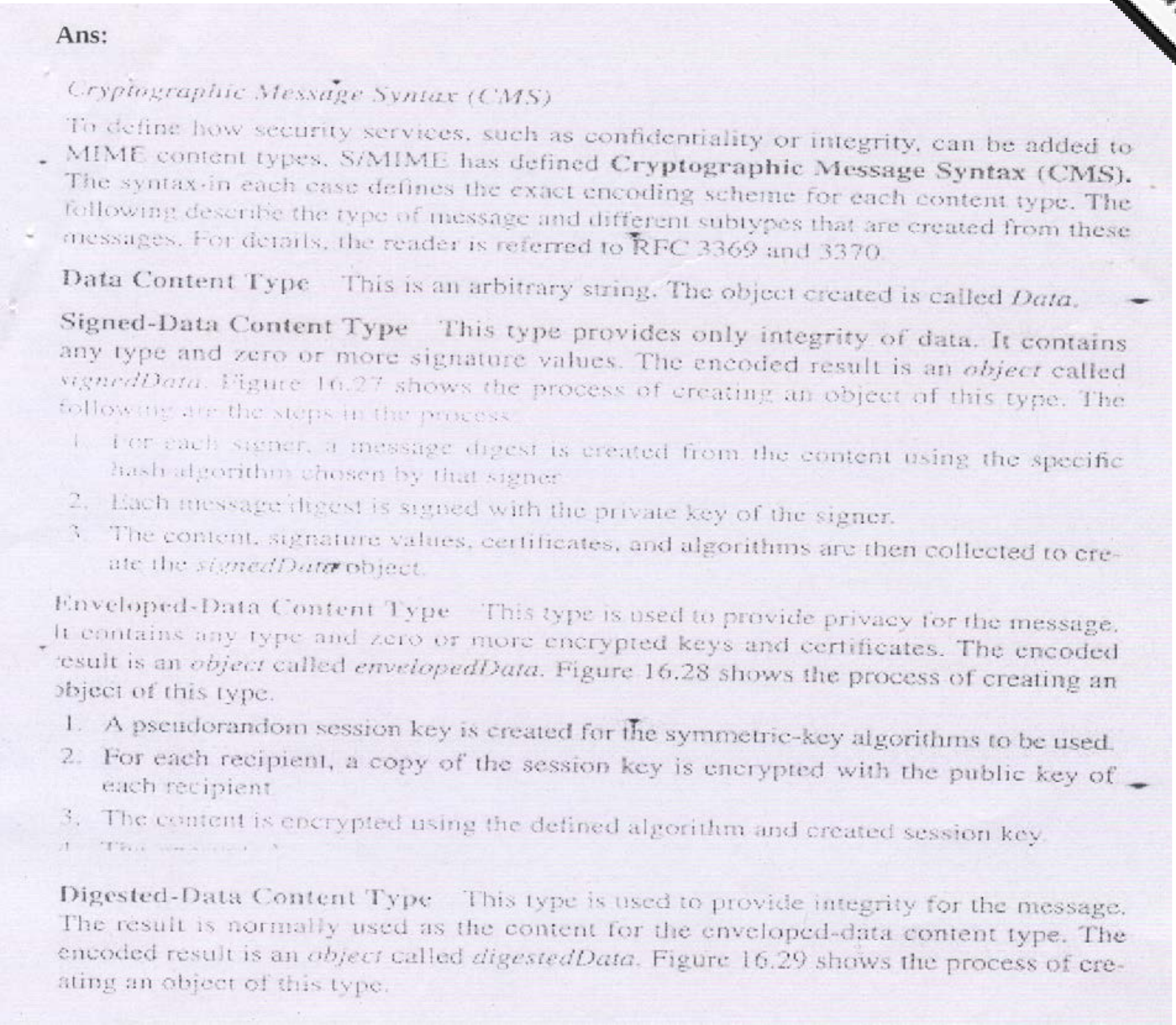
8. Bob now has the message *m*. In order to authenticate it he uses Alice's public key *Ae* to decrypt the signature *c* and hashes the message *m* using SHA-1.

If $SHA(m) = pk.decrypt_{Ae}(c)$

Then the message is authenticated.

b. What is CMS? Name all the content types defined by CMS and their purposes.

Answer:



Ans:

Cryptographic Message Syntax (CMS)

To define how security services, such as confidentiality or integrity, can be added to MIME content types, S/MIME has defined **Cryptographic Message Syntax (CMS).** The syntax in each case defines the exact encoding scheme for each content type. The following describe the type of message and different subtypes that are created from these messages. For details, the reader is referred to RFC 3369 and 3370.

Data Content Type This is an arbitrary string. The object created is called *Data.*

Signed-Data Content Type This type provides only integrity of data. It contains any type and zero or more signature values. The encoded result is an *object* called *signedData*. Figure 16.27 shows the process of creating an object of this type. The following are the steps in the process.

1. For each signer, a message digest is created from the content using the specific hash algorithm chosen by that signer.
2. Each message digest is signed with the private key of the signer.
3. The content, signature values, certificates, and algorithms are then collected to create the *signedData* object.

Enveloped-Data Content Type This type is used to provide privacy for the message. It contains any type and zero or more encrypted keys and certificates. The encoded result is an *object* called *envelopedData*. Figure 16.28 shows the process of creating an object of this type.

1. A pseudorandom session key is created for the symmetric-key algorithms to be used.
2. For each recipient, a copy of the session key is encrypted with the public key of each recipient.
3. The content is encrypted using the defined algorithm and created session key.
4. The [illegible]

Digested-Data Content Type This type is used to provide integrity for the message. The result is normally used as the content for the enveloped-data content type. The encoded result is an *object* called *digestedData*. Figure 16.29 shows the process of creating an object of this type.

<u>Text Book</u>

Behrouz A. Forouzan, Cryptography & Network Security, Special Indian Edition