III/IV B.Tech (Regular) DEGREE EXAMINATION

July/August, 2023 **Common to CB,CS,DS & IT Branches**

Sixth Semester **Cryptography & Network Security**

Time: Three Hours **Maximum:**70 Marks

Answer question 1 compulsory. **(14X1 = 14Marks)**

Answer one question from each unit. **(4X14=56 Marks)**

1.a) What is security?

Information needs to be hidden from unauthorizedaccess (confidentiality), protected from unauthorized change (integrity),and available to an authorized entity when it is needed (availability).

b) What are the securities Goals to protect the message?

Our three goals of security are confidentiality, integrity, and availability.

c) Why S-Boxes are required in DES?

An S-box is a substitution box and it is the only non-linear component in the cipher. Its main purpose is to obscure the relationship between the key, the plaintext, and the ciphertext**.**

d) Write the cipher text to the plain text "BEC CSE" using Rail Fence transposition cipher.

B		**C**		**S**	
	E		**C**		**E**

BCSECE

e) What is multiple DES?

DES was susceptible to attacks due to tremendous advances in computer hardware. Since DES was a very competent algorithm it would be feasible to reuse DES rather than writing a new cryptographic algorithm.

Due to this variations of DES were introduced known as multiple DES which were as follows:

- **Double DES**
- **Triple DES**

f) What is asymmetric cryptography?

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related keys -- one public key and one private key -- to encrypt and decrypt a message and protect it from unauthorized access or use.

g) How can validate the data authentication?

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

h) What is message authentication?

Message authentication allows to determine if a message has been altered in transmission between the session partners. A code is attached to each message by the sender and verified by the session partner.

i)What are digital Signature standards?

Digital Signature Standard (DSS) is a Federal Information Processing Standard(FIPS) which defines algorithms that are used to generate digital signatures with the help of Secure Hash Algorithm(SHA) for the authentication of electronic documents. DSS only provides us with the digital signature function and not with any encryption or key exchanging strategies.

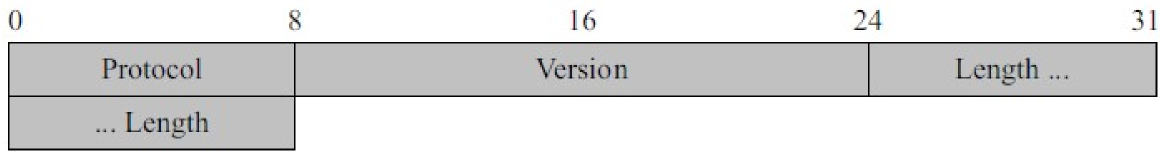
j) What is PGP?

- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
- PGP provides authentication through the use of Digital Signature.

k) List out the attacks on digital signatures.

- Key-Only Attack
- Known-Message Attack
- Chosen-Message Attack

l)Draw SSL Message format for record protocol?



Protocol-This 1-byte field defines the source or destination of the encapsulatedmessage. It is used for multiplexing and de-multiplexing.
Version-This 2-byte field defines the version of the SSL; one byte is the majorversion and the other is the minor. The current version of SSL is 3.0.
Length-This 2-byte field defines the size of the message (without the header)in bytes.

m) Write a short note on security association.

Security Association is a very important aspect of IPSec. IPSec requires a logical relationship,called a Security Association (SA), between two hosts. A Security Association is a contract between two parties; it creates a secure channel between them.

n) List out the two modes in security at the network layer.

- Transport mode
- Tunnel mode

Unit-I

2 (a) Encrypt the message 'this is exercise' using the following ciphers. Ignore the spaces between the words while encrypting. 7M

(a) Caesar cipher using the Key=5-------------------3M

(b) Play fair using the key='Monarchy'-------------4M

a) Caesar cipher for "***this is exercise***".

Key=5

So we shift every given letter by 5times such that it gives the encrypted text as follows.

T	h	i	s	i	s	e	x	e	r	c	i	s	e
Y	m	n	x	n	x	j	c	j	w	h	n	x	j

b) Play fair cipher using the key='***Monarchy***'

The best-known multiple-letter encryption cipher is the Playfair, which treats diagrams in the plaintext as single units and translates these units into ciphertext digrams.The Playfair algorithm is based on the use of a 5 * 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's Have His Carcase:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

In this case, the keyword is monarchy. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.

4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

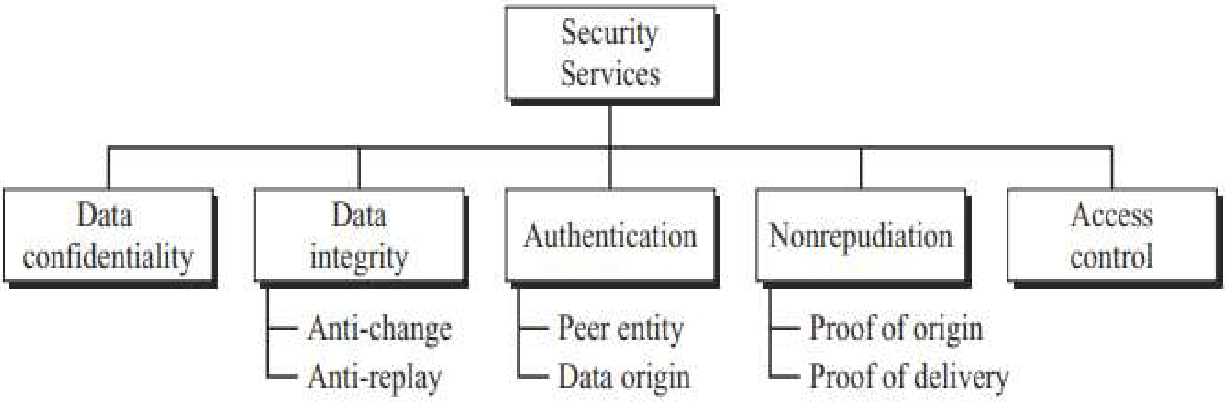
So by using the key monarchy we get the cipher text as PDSXSXIUKMBELI.

(b) Discuss in detail about the security services and mechanisioms. 7M

<u>Security services-----3M Security Mechanisms----4M</u>

ITU-T (X.800) has defined five services related to the security goals and attacks.

security services



Data confidentiality is designed to protect data from disclosure attack. The service as defined by X.800 is very broad and encompasses confidentiality of the whole message or part of a message and also protection against traffic analysis. That is, it is designed to prevent snooping and traffic analysis attack.

Data Integrity is designed to protect data from modification, insertion, deletion, and replaying by an adversary. It may protect the whole message or part of the message.

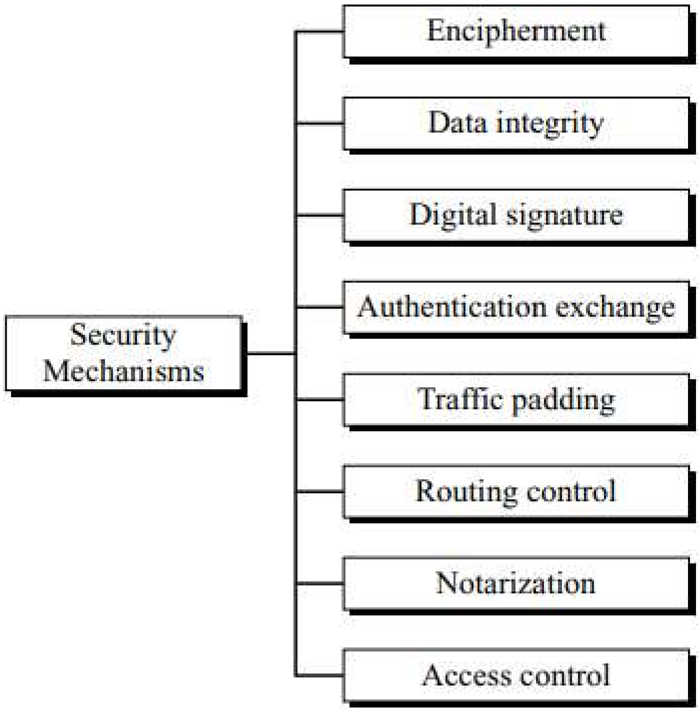
Authentication This service provides the authentication of the party at the other end of the line. In connection-oriented communication, it provides authentication of the sender or receiverduring the connection establishment. In connectionless communication, it authenticates the source of the data.

Nonrepudiation service protects against repudiation by either the sender or the receiver of the data. In nonrepudiation with proof of the origin, the receiver of the data can later prove the identity of the sender if denied. In nonrepudiation with proof of delivery, the sender of data can later prove that data were delivered to the intended recipient.

Access control provides protection against unauthorized access to data. The term access in this definition is very broad and can involve reading, writing, modifying, executing programs, and so on.

Security Mechanisms

ITU-T (X.800) also recommends some security mechanisms to provide the security services.



Encipherment, hiding or covering data, can provide confidentiality. It can also be used to complement other mechanisms to provide other services. Today two techniques□ cryptography and steganography□are used for enciphering.

The **data integrity** mechanism appends to the data a short checkvalue that has been created by a specific process from the data itself. The receiver receives the data and the checkvalue. He creates a new checkvalue from the received data and compares the newly created checkvalue with the one received. If the two checkvalues are the same, the integrity of data has been preserved.

Digital Signature A digital signature is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature. The sender uses a process that involves showing that she owns a private key related to the public key that she has announced publicly. The receiver uses the sender's public key to prove that the message is indeed signed by the sender who claims to have sent the message.

Authentication Exchange In authentication exchange, two entities exchange some messages to prove their identity to each other. For example, one entity can prove that she knows a secret that only she is supposed to know.

Traffic Padding means inserting some bogus data into the data traffic to thwart the adversary's attempt to use the traffic analysis.

Routing control means selecting and continuously changing different available routes between the sender and the receiver to prevent the opponent from eavesdropping on a particular route.

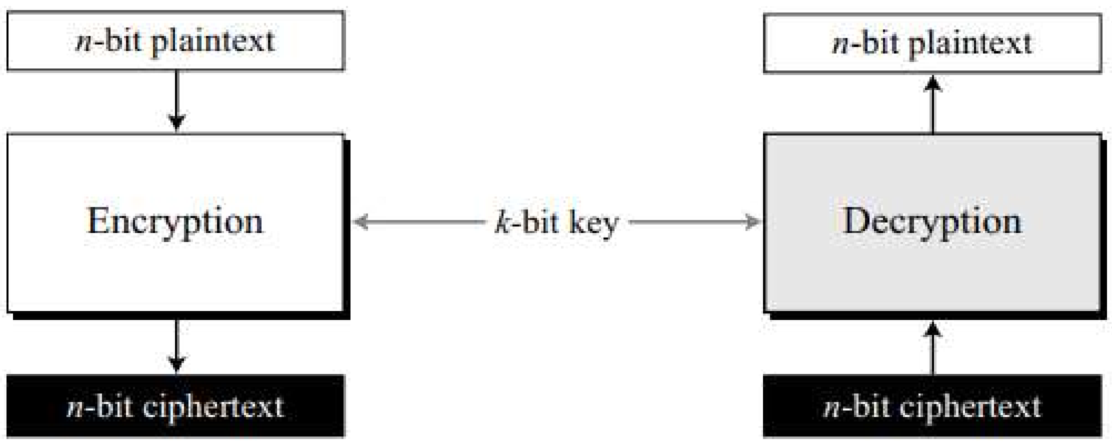
Notarization means selecting a third trusted party to control the communication between two entities. This can be done, for example, to prevent repudiation. The receiver can involve a trusted party to store the sender request in order to prevent the sender from later denying that she has made such a request.

Access control uses methods to prove that a user has access right to the data or resources owned by a system. Examples of proofs are passwords and PINs.

3. Describe the modes of modern Block Ciphers with design principles.

A symmetric-key modern block cipher encrypts an n-bit block of plaintext or decrypts an n-bit block of ciphertext. The encryption or decryption algorithm uses a k-bit key. The decryption algorithm must be the inverse of the encryption algorithm, and both operations must use the same secret key so that Bob can retrieve the message sent by Alice.

A modern block cipher



If the message has fewer than n bits, padding must be added to make it an n-bit block; if the message has more than n bits, it should be divided into n-bit blocks and the appropriate padding must be added to the last block if necessary. The common values for n are 64, 128, 256, or 512 bits.

Substitution or Transposition

A modern block cipher can be designed to act as a substitution cipher or a transposition cipher. This is the same idea as is used in traditional ciphers, except that the symbols to be substituted or transposed are bits instead of characters.

If the cipher is designed as a substitution cipher, a 1-bit or a 0-bit in the plaintext can be replaced by either a 0 or a 1. This means that the plaintext and the ciphertext can have a different number of 1's. A 64-bit plaintext block of 12 0's and 52 1's can be encrypted to a ciphertext of 34 0's and 30 1's. If the cipher is designed as a transposition cipher, the bits are only reordered (transposed); there is the same number of 1's in the plaintext and in the ciphertext. In either case, the number of n-bit possible plaintexts or ciphertexts is 2n , because each of the n bits in the block can have one of the two values, 0 or 1.

Modern block ciphers are designed as substitution ciphers because the inherent characteristics of transposition (preserving the number of 1's or 0's) makes the cipher vulnerable to exhaustive-search attacks.

Block Ciphers as Permutation Groups

we need to know whether a modern block cipher is a group. To answer this question, first assume that the key is long enough to choose every possible mapping from the input to the output. Call this a full-size key cipher. In practice, however, the key is smaller; only some mappings from the input to the output are possible. Although a block cipher needs to have a key that is a secret between the sender and the receiver, there are also keyless components that are used inside a cipher.

Full-Size Key Ciphers

Although full-size key ciphers are not used in practice, we first discuss this category to make the discussion of partial-size key ciphers understandable.

Full-Size Key Transposition Block Ciphers

A full-size key transposition cipher only transposes bits without changing their values, so it can be modelled as an n-object permutation with a set of n! permutation tables in which the key defines which table is used by Alice and Bob. We need to have n! possible keys, so the key should have [log2 n!] bits.

Full-Size Key Substitution Block Ciphers

A full-size key substitution cipher does not transpose bits; it substitutes bits. At first glance, it appears that a full-size key substitution cipher cannot be modelled as a permutation. However, we can model the substitution cipher as a permutation if we can decode the input and encode the output. Decoding means transforming an n-bit integer into a 2n -bit string with only a single 1 and 2n − 1 0's. The position of the single 1 is the value of the integer, in which the positions range from 0 to 2n − 1. Encoding is the reverse process. Because the new input and output have always a single 1, the cipher can be modelled as a permutation of 2n! objects.

Unit-II

4 a) Perform the RSA algorithm on the given data and explain how encryption and decryption are performed on the message: p=17, q=31; e=7; M=2.

Step 1. Calculate N which is a product of two distinct prime numbers *p* and *q*
p = 17
q = 31
Calculate N 7*31=527

Step 2. Find θ(N) which is *(p-1) * (q-1)*
θ(N) = (17-1) * (31-1) = 480
Step 3. Select *e* such that gcd(θ(N),*e*) = 1 and 1 < *e* < θ(N)

Possible Values of e
7,11,13,17,19,23,29,31,37,41,43,47,49,53,59,61,67,71,73,77,79,83,89,91,97,101,103,107,109,113,119,121,127,131,133,137,139,143,149,151,157,161,163,167,169,173,179,181,187,191,193,197,199,203,209,211,217,221,223,227,229,233,239,241,247,251,253,257,259,263,269,271,277,281,283,287,289,293,299,301,307,311,313,317,319,323,329,331,337,341,343,347,349,353,359,361,367,371,373,377,379,383,389,391,397,401,403,407,409,413,419,421,427,431,433,437,439,443,449,451,457,461,463,467,469,473,479

Enter e =7

Step 4. Calculate *d* such that d*e mod(θ(N) = 1
d = 343

tep 5. Set public key and private key
Public Key {e , n} = 7,527
Private Key {d , n} = 343,527
Step 6. Enter plaintext message M to encrypt such that M < N (C = M^d (mod n))
Enter a number to encrypt M=2

Encryption = $C = M^e \bmod n$

= 2^7 mod 527

=128 mod 527

=128

Decryption = $M = C^d \bmod n$

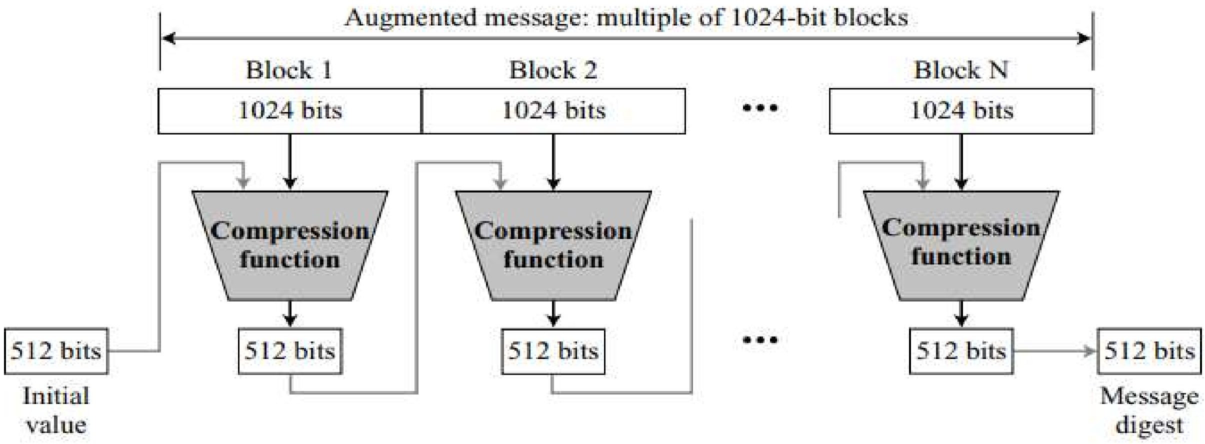
=128^{343} *mod 527*

=*2*

b) Discuss the process of the message digest generation using SHA 512. 7M

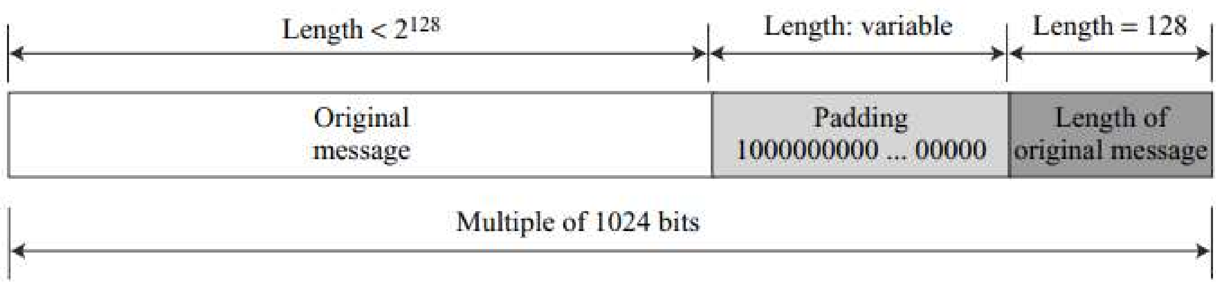
<u>Diagram----2M Explanation-----5M</u>

SHA-512 is the version of SHA with a 512-bit message digest.SHA-512 creates a digest of 512 bits from a multiple-block message. Each block is 1024 bits in length, as shown in Figure.



The digest is initialized to a predetermined value of 512 bits. The algorithm mixes this initial value with the first block of the message to create the first intermediate message digest of 512 bits. This digest is then mixed with the second block to create the second intermediate digest. Finally, the (N − 1)th digest is mixed with the Nth block to create the Nth digest.When the last block is processed, the resulting digest is the message digest for the entire message. Message Preparation SHA-512 insists that the length of the original message be less than 2128 bits. This means that if the length of a message is equal to or greater than 2128, it will not be processed by SHA-512. This is not usually a problem because 2128 bits is probably larger than the total storage capacity of any system.

Length Field and Padding Before the message digest can be created, SHA-512 requires the addition of a 128-bit unsigned-integer length field to the message that defines the length of the message in bits. This is the length of the original message before padding. An unsigned integer field of 128 bits can define a number between 0 and 2128 − 1, which is the maximum length othe message allowed in SHA-512.



Before the addition of the length field, we need to pad the original message to make the length a multiple of 1024. The length of the padding field can be calculated as follows. Let |M| be the length of the original message and |P| be the length of the padding field. The format of the padding is one 1 followed by the necessary number of 0s.

Words SHA-512 operates on words; it is word oriented. A word is defined as 64 bits. This means that, after the padding and the length field are added to the message, each block of the message consists of sixteen 64-bit words. The message digest is also made of 64-bit words, but the message digest is only eight words, and the words are named A, B, C, D, E, F, G, and H.

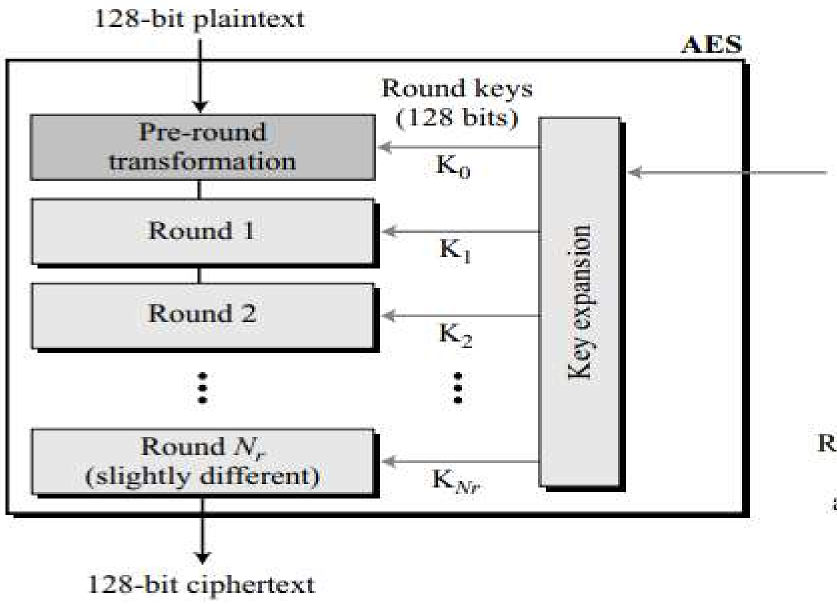
Word Expansion Before processing, each message block must be expanded. A block is made of 1024 bits, or sixteen 64-bit words. As we will see later, we need 80 words in the processing phase. So the 16-word block needs to be expanded to 80 words, from W0 to W79.

Compression Function SHA-512 creates a 512-bit (eight 64-bit words) message digest from a multiple-block message where each block is 1024 bits. The processing of each block of data in SHA512 involves 80 rounds. Figure 12.10 shows the general outline for the compression function. In each round, the contents of eight previous buffers, one word from the expanded block (Wi), and one 64-bit constant (Ki) are mixed together and then operated on to create a new set of eight buffers. At the beginning of processing, the values of the eight buffers are saved into eight temporary variables. At the end of the processing (after step 79), these values are added to the values created from step 7.

5 a) With a neat diagram illustrate the process of AES **7M**

<u>Diagram----2M Process ---------5M</u>

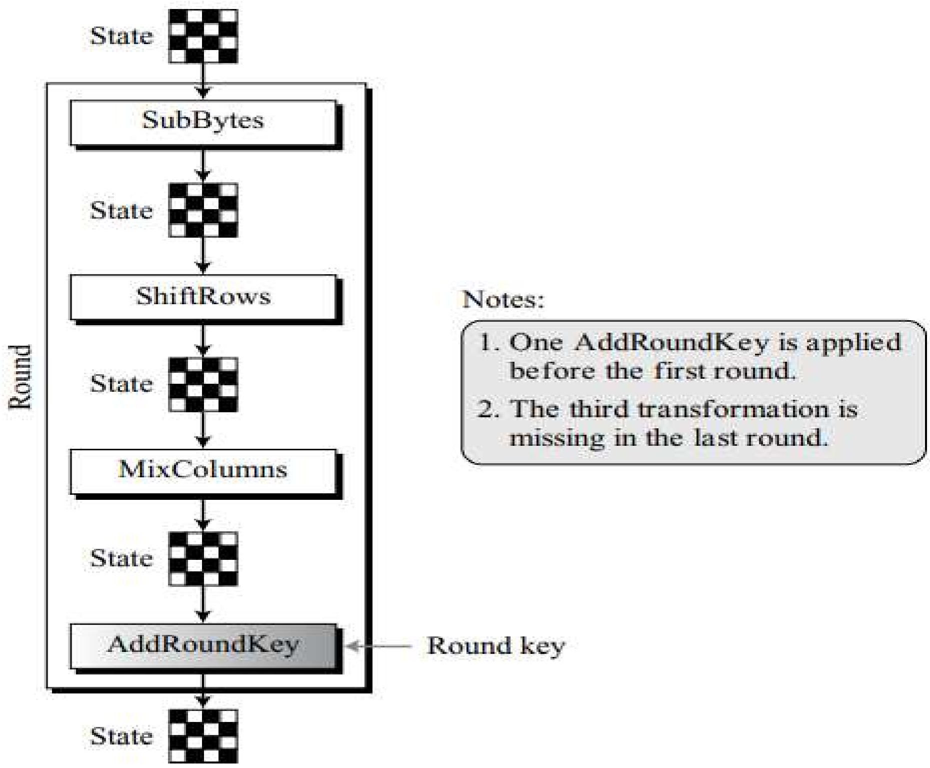
AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits. It uses 10, 12, or 14 rounds. The key size, which can be 128, 192, or 256 bits, depends on the number of rounds. The general design for the encryption algorithm (called cipher); the decryption algorithm (called inverse cipher) is similar, but the round keys are applied in the reverse order.



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

The structure of each round at the encryption side. Each round, except the last, uses four transformations that are invertible. The last round has only three transformations., each transformation takes a state and creates another state to be used for the next transformation or the next round. The pre-round section uses only one transformation (AddRoundKey); the last round uses only three transformations (MixColumns transformation is missing).



Sub Bytes

The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits. The left digit defines the row and the right digit defines the column of the substitution table. The two hexadecimal digits at the junction of the row and the column are the new byte. In the SubBytes transformation, the state is treated as a 4 × 4 matrix of bytes. Transformation is done one byte at a time. The contents of each byte is changed, but the arrangement of the bytes in the matrix remains the same. In the process, each byte is transformed independently. There are sixteen distinct byte-to-byte transformations.

ShiftRows

In the encryption, the transformation is called ShiftRows and the shifting is to the left.The number of shifts depends on the row number (0, 1, 2, or 3) of the state matrix. This means the row 0 is not shifted at all and the last row is shifted three bytes.

MixColumns

The MixColumns transformation operates at the column level; it transforms each column of the state to a new column. The transformation is actually the matrix multiplication of a state column by a constant square matrix. The bytes in the state column and constants matrix are interpreted as 8-bit words (or polynomials) with coefficients in GF(2). Multiplication of bytes is done in GF(28) with modulus (10001101) or (x8 + x4 + x3 + x + 1). Addition is the same as XORing of 8-bit words.

AddRoundKey

AddRoundKey also proceeds one column at a time. It is similar toMixColumns in this respect. MixColumns multiplies a constant square matrix by each state column; AddRoundKey adds a round key word with each state column matrix. The operation in MixColumns is matrix multiplication; the operation in AddRoundKey is matrix addition. Since addition and subtraction in this field are the same, the AddRoundKey transformation is the inverse of itself.

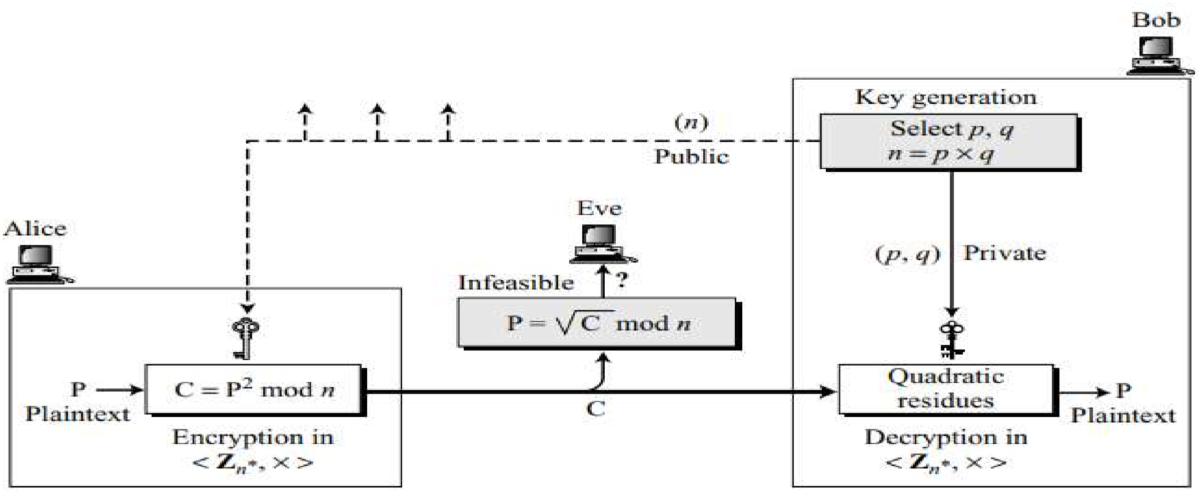
b) Write a short note on Robin Cryptosystem, and Elgamal Cryptosystem. 7M

Robin Cryptosystem----3M Elgamal Cryptosystem---4M

RABIN CRYPTOSYSTEM

The Rabin cryptosystem, devised by M. Rabin, is a variation of the RSA cryptosystem. RSA is based on the exponentiation congruence; Rabin is based on quadratic congruence. The Rabin cryptosystem can be thought of as an RSA cryptosystem in which the value of e and d are fixed; e = 2 and d = 1/2. In other words, the encryption is C ≡ P2 (mod n) and the decryption is P ≡ C1/2 (mod n). The public key in the Rabin cryptosystem is n; the private key is the tuple (p, q). Everyone can encrypt a message using n; only Bob can decrypt the message using p and q. Decryption of the message is infeasible for Eve because she does not know the values of p and q.

Encryption, decryption, and key generation in the Rabin cryptosystem



Rabin_Key_Generation

{

Choose two large primes p and q in the form $4k + 3$ and $p \neq q$.

$n \leftarrow p \times q$

Public_key ← n // To be announced publicly

Private_key ← (p, q) // To be kept secret

return Public_key and Private_key

}

Although the two primes, p and q, can be in the form 4k + 1 or 4k + 3, the decryption process becomes more difficult if the first form is used. It is recommended to use the second form, 4k + 3, to make the decryption for Alice much easier.

Encryption

Anyone can send a message to Bob using his public key.

```
Rabin_Encryption (n, P)          // n is the public key; P is the ciphertext from Zn*
{
   C ← P^2 mod n                 // C is the ciphertext
   return C
}
```

Although the plaintext P can be chosen from the set Zn, we have defined the set to be in Zn* to make the decryption easier. Encryption in the Rabin cryptosystem is very simple. The operation needs only one multiplication, which can be done quickly. This is beneficial when resources are limited. For example, smart cards have limited memory and need to use short CPU time.

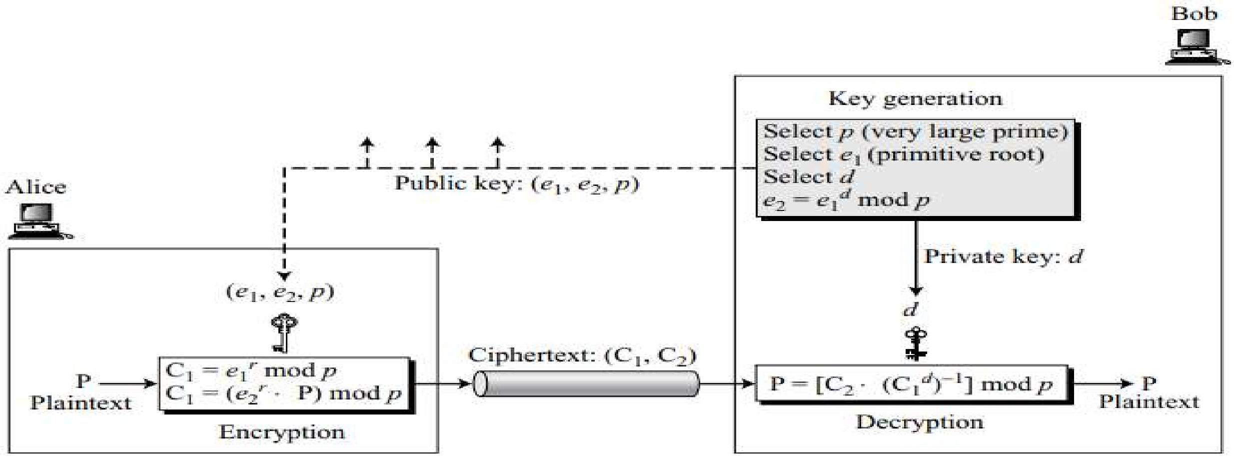
Decryption

Bob can use rabin decryption Algorithm to decrypt the received ciphertext.

```
Rabin_Decryption (p, q, C)          // C is the ciphertext; p and q are private keys
{
    a1 ← +(C^((p+1)/4)) mod p
    a2 ← −(C^((p+1)/4)) mod p
    b1 ← +(C^((q+1)/4)) mod q
    b2 ← −(C^((q+1)/4)) mod q
    // The algorithm for the Chinese remainder theorem is called four times.
    P1 ← Chinese_Remainder (a1, b1, p, q)
    P2 ← Chinese_Remainder (a1, b2, p, q)
    P3 ← Chinese_Remainder (a2, b1, p, q)
    P4 ← Chinese_Remainder (a2, b2, p, q)
    return P1, P2, P3, and P4
}
```

Several points should be emphasized here. The decryption is based on the solution of quadratic congruence, discussed in Chapter 9. Because the received ciphertext is the square of the plaintext, it is guaranteed that C has roots (quadratic residues) in Zn*. The Chinese remainder algorithm is used to find the four square roots. The most important point about the Rabin system is that it is not deterministic. The decryption has four answers. It is up to the receiver of the message to choose one of the four as the final answer.

ElGamal Cryptosystem



if p is a very large prime, e1 is a primitive root in the group G = and r is an integer, then e2 = e1 r mod p is easy to compute using the fast exponential algorithm (square-and-multiply method), but given e2, e1, and p, it is infeasible to calculate r = loge1e2 mod p.

```
ElGamal_Key_Generation
{
    Select a large prime p
    Select d to be a member of the group G = <Z_p*, ×> such that 1 ≤ d ≤ p − 2
    Select e1 to be a primitive root in the group G = <Z_p*, ×>
    e2 ← e1^d mod p
    Public_key ← (e1, e2, p)            // To be announced publicly
    Private_key ← d                     // To be kept secret
    return Public_key and Private_key
}
```

Encryption Anyone can send a message to Bob using his public key. The encryption process is shown in Algorithm below. If the fast exponential algorithmis used, encryption in the ElGamal cryptosystem can also be done in polynomial time complexity.

```
ElGamal_Encryption (e1, e2, p, P)          // P is the plaintext
{
    Select a random integer r in the group G = <Z_p*, ×>
    C1 ← e1^r mod p
    C2 ← (P × e2^r) mod p                  // C1 and C2 are the ciphertexts
    return C1 and C2
}
```

```
ElGamal_Decryption (d, p, C1, C2)          // C1 and C2 are the ciphertexts
{
    P ← [C2 (C1^d)^-1] mod p               // P is the plaintext
    return P
}
```

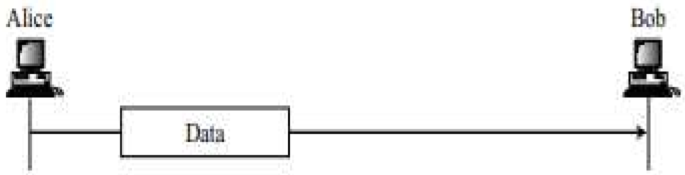
Unit-III

6 a) Explain five security services of PGP. 7M

PGP was invented by Phil Zimmermann to provide e-mail with privacy, integrity, and authentication. PGP can be used to create a secure e-mail message or to store a file securely for future retrieval.

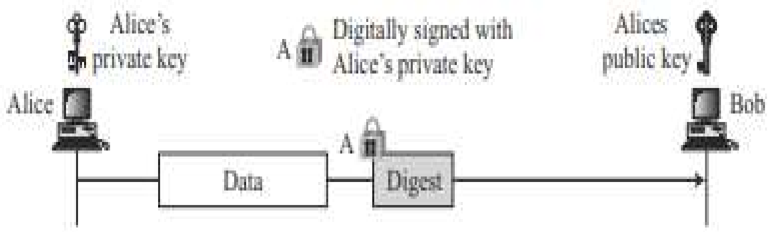
Plaintext

The simplest scenario is to send the e-mail message (or store the file) in plaintext as shown in Figure. There is no message integrity or confidentiality in this scenario. Alice, the sender, composes a message and sends it to Bob, the receiver. The message is stored in Bob's mailbox until it is retrieved by him.



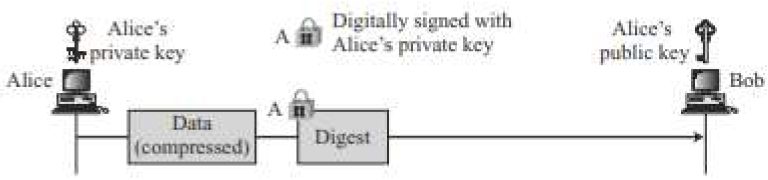
Message Integrity

Probably the next improvement is to let Alice sign the message. Alice creates a digest of the message and signs it with her private key. When Bob receives the message, he verifies the message by using Alice's public key. Two keys are needed for this scenario. Alice needs to know her private key; Bob needs to know Alice's public key. Figure shows the situation.



Compression

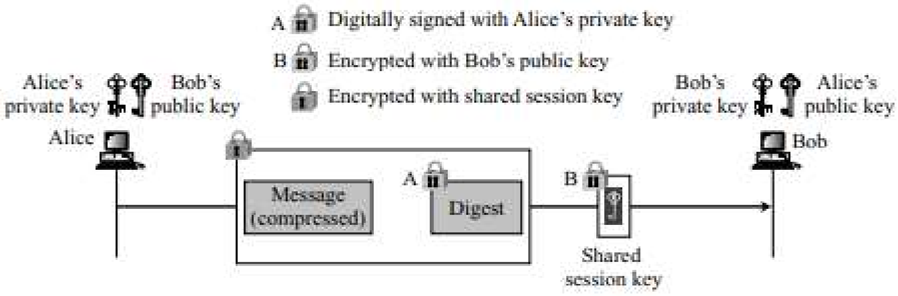
A further improvement is to compress the message to make the packet more compact. This improvement has no security benefit, but it eases the traffic. Figure shows the new scenario.



Confidentiality with One-Time Session Key

As we discussed before, confidentiality in an e-mail system can be achieved using conventional encryption with a one-time session key. Alice can create a session key, use the session key to encrypt the message and the digest, and send the key itself with the message. However, to protect the session key, Alice encrypts it with Bob's public key.

When Bob receives the packet, he first decrypts the key, using his private key to remove the key. He then uses the session key to decrypt the rest of the message. After decompressing the rest of the message, Bob creates a digest of the message and checks to see if it is equal to the digest sent by Alice. If it is, then the message is authentic.



Code Conversion

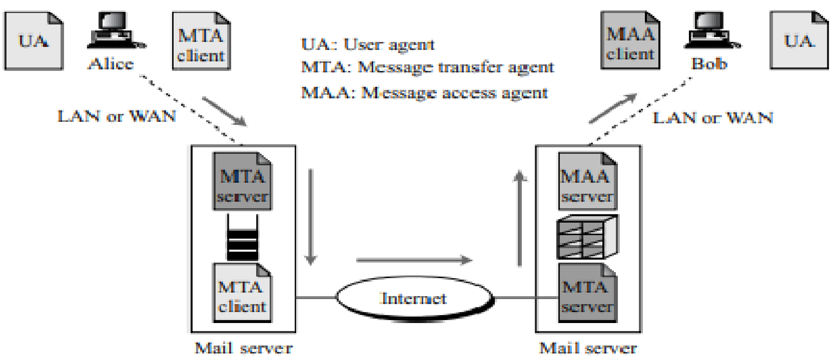
Another service provided by PGP is code conversion. Most e-mail systems allow the message to consist of only ASCII characters. To translate other characters not in the ASCII set, PGP uses Radix-64 conversion. Each character to be sent (after encryption) is converted to Radix-64 code.

6 b)Explain e-mail system architecture? 7M

<u>Diagram----2M Explanation----5M</u>

E-mail Architecture

The most common scenario in a one-way e-mail exchange. Assume that Alice is working in an organization that runs an e-mail server; every employee is connected to the e-mail server through a LAN. Or alternatively, Alice could



E-mail architecture

The administrator of the e-mail server at Alice's site has created a queuing system that sends e-mail to the Internet one by one. The administrator of the e-mail server at Bob's site has created a mailbox for every user connected to the server; the mailbox holds the received messages until they are retrieved by the recipient.

When Alice needs to send a message to Bob, she invokes a **user agent(UA)** program to prepare the message. She then uses another program, a **message transfer agent (MTA),** to send the message to the mail server at her site. Note that the MTA is a client/server program with the client installed at

The message received at the mail server at Alice's site is queued with all other messages; each goes to its corresponding destination. In Alice's case, her message goes to the mail server at Bob's site. A client/server MTA is responsible for the e-mail transfer between the two servers. When the message arrives at the destination mail server, it is stored in Bob's mailbox, a special file that holds the message until it is retrieved by Bob.When Bob needs to retrieve his messages, including the one sent by Alice, he invokes another program, which we call a message access agent (MAA). The MAA is also designed as a client/server program with the client installed at Bob's computer and the server installed at the mail server.

There are several important points about the architecture of the e-mail system.

a. The sending of an e-mail from Alice to Bob is a store-retrieve activity. Alice can send an e-mail today.Bob, being busy, may check his e-mail three days later. During this time, the e-mail is stored in Bob'smailbox until it is retrieved.

b. The main communication between Alice and Bob is through two application programs: the MTA client at Alice's. computer and the MAA client at Bob's computer.

c. The MTA client program is a push program; the client pushes the message when Alice needs to send it.The client program is a pull program; the client pulls the messages when Bob is ready to retrieve his e- mail.

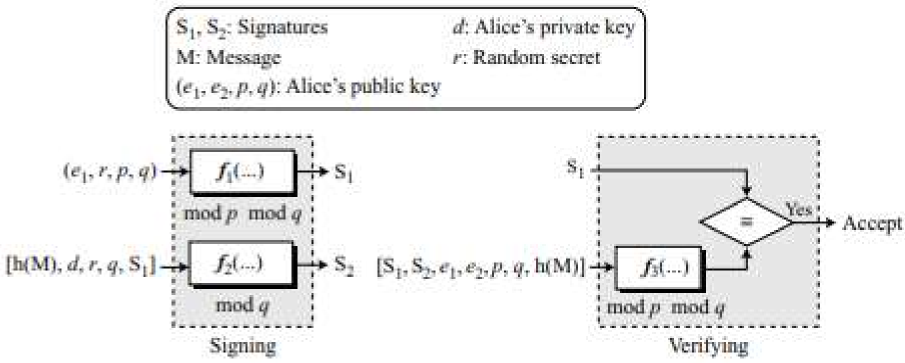
d. Alice and Bob cannot directly communicate using an MTA client at the sender site and an MTA server atthe receiver site. This requires that the MTA server be running all the time, because Bob does not knowwhen a message will arrive. This is not practical, because Bob probably turns off his computer when he does not need it.

7 a) Explain the concept of digital Signature with a neat diagram. 7M

Diagrams---2M Explanation---5M

Digital signature.

The Digital Signature Standard (DSS) was adopted by the National Institute of Standards and Technology (NIST) in 1994. NIST published DSS as FIPS 186. DSS uses a digital signature algorithm (DSA) based on the ElGamal scheme with some ideas from the Schnorr scheme. DSS has been criticized from the time it was published. The main complaint regards the secrecy of DSS design. The second complaint regards the size of the prime, 512 bits. Later NIST made the size variable to respond to this complaint. gives the general idea behind the DSS scheme.



General idea behind DSS scheme

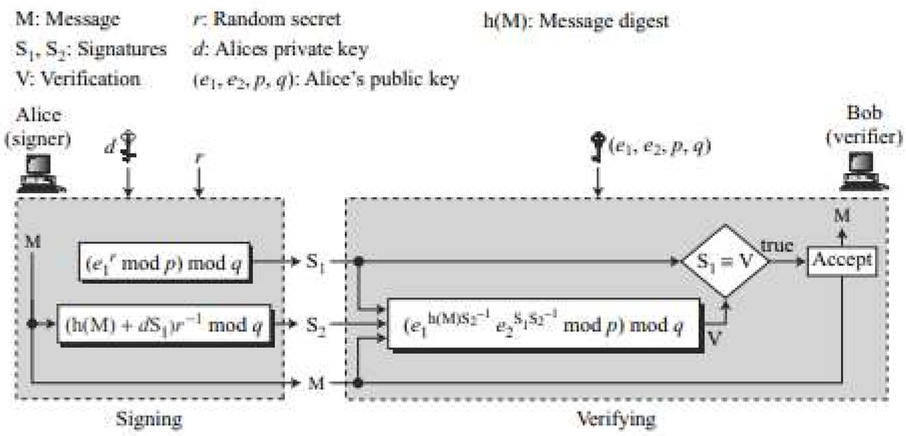
In the signing process, two functions create two signatures; in the verifying process, the output of one function is compared to the first signature for verification. This is similar to Schnorr, but the inputs are different. Another difference is that this scheme uses the message digest (not the message) as part of inputs to functions 1 and 3. The interesting point is that the scheme uses two public moduli: p and q. Functions 1 and 3 use both p and q; function 2 uses only q. The details of inputs and the functions will be discussed shortly.

Key Generation

Before signing a message to any entity, Alice needs to generate keys and announce the public ones to the public.

1. Alice chooses a prime p, between 512 and 1024 bits in length. The number of bits in p must be a multiple of 64.
2. Alice chooses a 160-bit prime q in such a way that q divides (p − 1).
3. Alice uses two multiplication groups and ; the second is a sub group of the first.
4. Alice creates e1 to be the qth root of 1 modulo p (e1 p = 1 mod p). To do so, Alice chooses a primitive element in Zp, e0, and calculates e1 = e0 (p−1)/q mod p.
5. Alice chooses d as the private key and calculates e2 = e1 d mod p.
6. Alice's public key is (e1, e2, p, q); her private key is (d).

Verifying and Signing



DSS Scheme

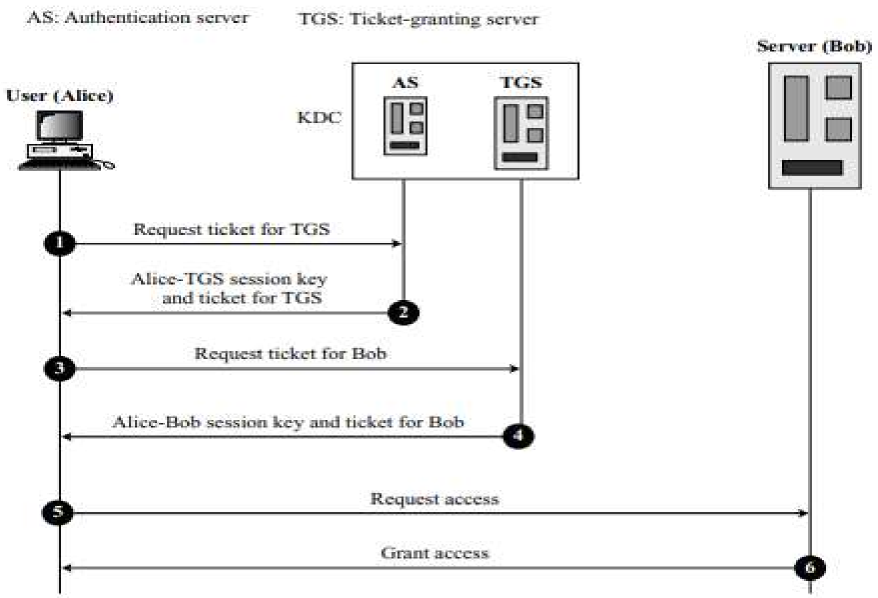
SigningThe following shows the steps to sign the message:
1. Alice chooses a random number r ($1 \le r \le q$). Note that although public and private keys can be chosen once and used to sign many messages, Alice needs to select a new r each time she needs to sign a new message
2. Alice calculates the first signature S1 = (e1 r mod p) mod q. Note that the value of the first signature does not depend on M, the message.
3. Alice creates a digest of message h(M).
4. Alice calculates the second signature S2 = (h(M) + d S1)r −1mod q. Note that the calculation of S2 is done in modulo q arithmetic.
5. Alice sends M, S1, and S2 to Bob.

VerifyingFollowing are the steps used to verify the message when M, S1, and S2 are received:
1. Bob checks to see if $0 < S1 < q$.
2. Bob checks to see if $0 < S2 < q$.
3. Bob calculates a digest of M using the same hash algorithm used by Alice.
4. Bob calculates V = [(e1 h(M)S2 −1 e2 S1S2 −1) mod p] mod q.
5. If S1 is congruent to V, the message is accepted; otherwise, it is rejected.

7 b) Illustrate functioning process of Kerberos. 7M

<u>Diagram----2M Explanation-----5M</u>
Kerberos is an authentication protocol, and at the same time a KDC, that has become very popular. Several systems, including Windows 2000, use Kerberos. It is named after the three-headed dog in Greek mythology that guards the gates of Hades. Originally designed at MIT, it has gone through several versions. We only discuss version 4, the most popular, and we briefly explain the difference between version 4 and version 5 (the latest).

Servers
Three servers are involved in the Kerberos protocol: an authentication server (AS), a ticket-granting server (TGS), and a real (data) server that provides services to others. In our examples and figures, Bob is the real server and Alice is the user requesting service. Figure 15.7 shows the relationship between these three servers.



Kerberos servers

Authentication Server (AS)

The **authentication server(AS)** is the KDC in the Kerberos protocol. Each user registers with the AS and is granted a user identity and a password. The AS has a database with these identities and the corresponding passwords. The AS verifies the user, issues a session key to be used between Alice and the TGS, and sends a ticket for the TGS.

Ticket-Granting Server (TGS)

The **ticket-granting server (TGS)** issues a ticket for the real server (Bob). It also provides the session key (KAB) between Alice and Bob. Kerberos has separated user verification from the issuing of tickets. In this way, though Alice verifies her ID just once with the AS, she can contact the TGS multiple times to obtain tickets for different real servers.

Real Server

The real server (Bob) provides services for the user (Alice). Kerberos is designed for a client-server program, such as FTP, in which a user uses the client process to access the server process. Kerberos is not used for person-to-person authentication.

Operation

A client process (Alice) can access a process running on the real server (Bob) in six steps,

1. Alice sends her request to the AS in plain text using her registered identity.

2. The AS sends a message encrypted with Alice's permanent symmetric key, KA-AS. The message contains two items: a session key, KA-TGS, that is used by Alice to contact the TGS, and a ticket for the TGS that is encrypted with the TGS symmetric key, KAS-TGS. The password and the appropriate algorithm together create KA-AS if the password is correct. The password is then immediately destroyed; it is not sent to the network and it does not stay in the terminal. It is used only for a moment to create KA-AS. The process now uses KA-AS to decrypt the message sent. KA-TGS and the ticket are extracted

3. Alice now sends three items to the TGS. The first is the ticket received from the AS. The second is the name of thereal server (Bob), the third is a timestamp that is encrypted by KA-TGS. The timestamp prevents a replay by Eve.

4. The TGS sends two tickets, each containing the session key between Alice and Bob, KA-B. The ticket for Alice is encrypted with KA-TGS; the ticket for Bob is encrypted with Bob's key, KTGS-B. Note that Eve cannot extract KAB because Eve does not know KA-TGS or KTGS-B. She cannot replay step 3 because she cannot replace the timestamp with a new one (she does not know KA-TGS).

5. Alice sends Bob's ticket with the timestamp encrypted by KA-B.

6. Bob confirms the receipt by adding 1 to the timestamp. The message is encrypted with KA-B and sent to Alice.

UNIT-4

8 a) Illustrate the SSL services with a neat architecture along with the four protocols.14M

SSL architecture----2M

SSL Services---------4M
Four Protocols each----2M

SSL ARCHITECTURE

SSL is designed to provide security and compression services to data generated from the application layer. Typically, SSL can receive data from any application layer protocol, but usually the protocol is HTTP. The data received from the application is compressed (optional), signed, and encrypted. The data is then passed to a reliable transport layer protocol such as TCP.

Services

SSL provides several services on data received from the application layer.

Fragmentation

First, SSL divides the data into blocks of 214 bytes or less.

Compression

Each fragment of data is compressed using one of the lossless compression methods negotiated between the client and server. This service is optional.

Message Integrity

To preserve the integrity of data, SSL uses a keyed-hash function to create a MAC.

Confidentiality

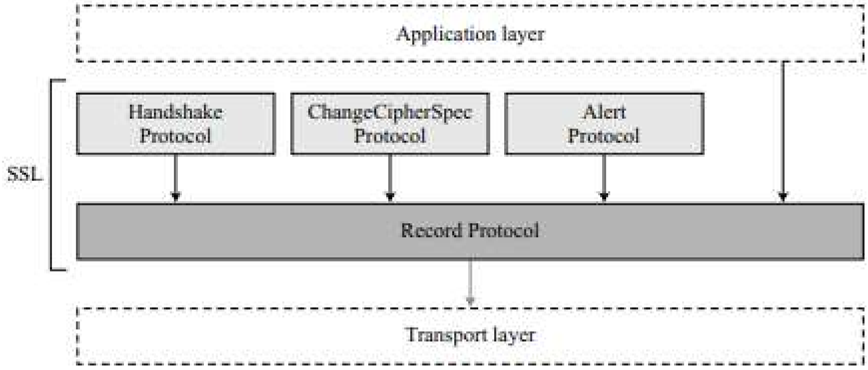
To provide confidentiality, the original data and the MAC are encrypted using symmetric key cryptography.

Framing

A header is added to the encrypted payload. The payload is then passed to a reliable transport layer protocol.

FOUR PROTOCOLS

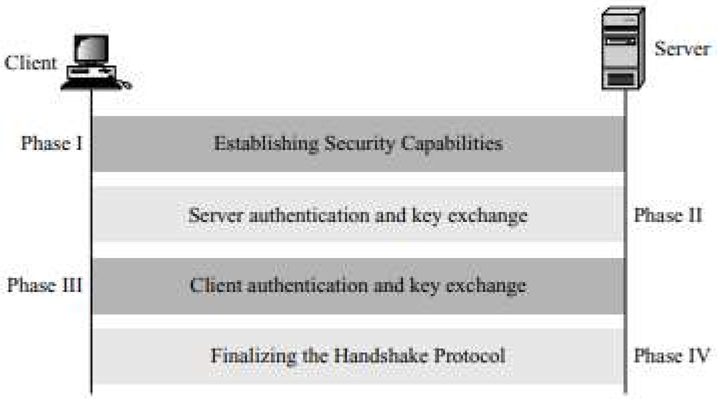
We have discussed the idea of SSL without showing how SSL accomplishes its tasks. SSL defines four protocols in two layers. The Record Protocol is the carrier. It carries messages from three other protocols as well as the data coming from the application layer. Messages from the Record Protocol are payloads to the transport layer, normally TCP. The Handshake Protocol provides security parameters for the Record Protocol. It establishes a cipher set and provides keys and security



Four SSL protocols

Handshake Protocol

The **Handshake Protocol** uses messages to negotiate the cipher suite, to authenticate the server to the client and client to the server if needed, and to exchange information for building the cryptographic secrets. The handshaking is done in four phases



Handshake Protocol

Phase I: Establishing Security Capability

In Phase I, the client and the server announce their security capabilities and choose those that are convenient for both. In this phase, a session ID is established and the cipher suite is chosen. The parties agree upon a particular compression method. Finally, two random numbers are selected, one by the client and one by the server, to be used for creating a master secret as we saw before. Two messages are exchanged in this phase: Client Hello and Server Hello messages

Client Hello The client sends the Client Hello message. It contains the following:

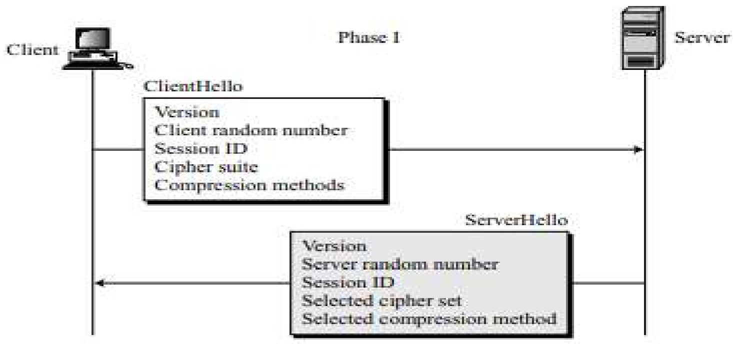
a. The highest SSL version number the client can support.

b.A 32-byte random number (from the client) that will be used for master secret generation.

c. session ID that defines the session.

d.A cipher suite that defines the list of algorithms that the client can support.

e. A list of compression methods that the client can support.



Phase I: Establishing Security Capability

Server Hello The server responds to the client with a Server Hello message. It contains the following:

a. An SSL version number. This number is the lower of two version numbers: the highest supported by the client and the highest supported by the server.

b. A 32-byte random number (from the server) that will be used for master secret generation.

c. A session ID that defines the session.

d. The selected cipher set from the client list.

e. The selected compression method from the client list

Phase II: Server Key Exchange and Authentication

In phase II, the server authenticates itself if needed. The sender may send its certificate, its public key, and may also request certificates from the client. At the end, the server announces that the server Hello process is done.

Phase II: Server Key Exchange and Authentication

Certificate If it is required, the server sends a Certificate message to authenticate itself. The message includes a list of certificates of type X.509. The certificate is not needed if the key-exchange algorithm is anonymous Diffie-Hellman.

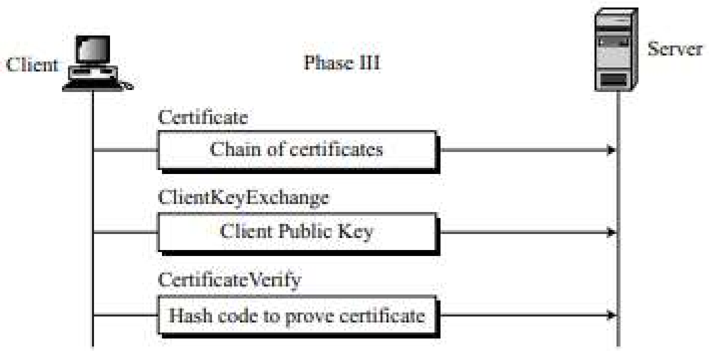
Server Key Exchange After the Certificate message, the server sends a Server KeyExchange message that includes its contribution to the pre-master secret. This message is not required if the key-exchange method is RSA or fixed Diffie-Hellman.

Certificate Request The server may require the client to authenticate itself. In this case, the server sends a Certificate Request message in Phase II that asks for certification in Phase III from the client. The server cannot request a certificate from the client if it is using anonymous Diffie-Hellman.

Server Hello Done The last message in Phase II is the Server Hello Done message, which is a signal to the client that Phase II is over and that the client needs to start Phase III

Phase III: Client Key Exchange and Authentication

Phase III is designed to authenticate the client. Up to three messages can be sent from the client to the server



Phase III: Client Key Exchange and Authentication

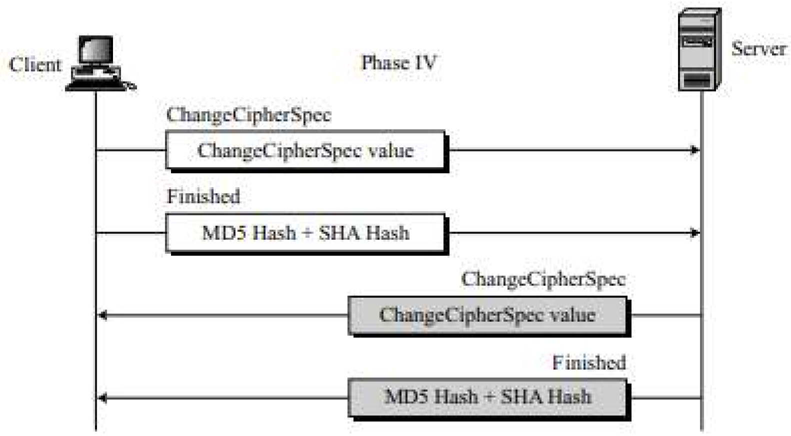
Certificate To certify itself to the server, the client sends a Certificate message. Note that the format is the same as the Certificate message sent by the server in Phase II, but the contents are different. It includes the chain of certificates that certify the client .This message is sent only if the server has requested a certificate in Phase II. If there is a request and the client has no certificate to send, it sends an Alert message (part of the Alert Protocol to be discussed later) with a warning that there is no certificate. The server may continue with the session or may decide to abort.

Client Key Exchange After sending the Certificate message, the client sends a Client Key Exchange message, which includes its contribution to the pre-master secret. The contents of this message are based on the key-exchange algorithm used. If the method is RSA, the client creates the entire pre-master secret and encrypts it with the RSA public key of the server. If the method is anonymous or ephemeral Diffie-Hellman, the client sends its Diffie-Hellman half-key. If the method is Fortezza, the client sends the Fortezza parameters. The contents of this message are empty if the method is fixed Diffie-Hellman.

Certificate Verify If the client has sent a certificate declaring that it owns the public key in the certificate, it needs to prove that it knows the corresponding private key. This is needed to thwart an impostor who sends the certificate and claims that it comes from the client. The proof of private-key possession is done by creating a message and signing it with the private key. The server can verify the message with the public key already sent to ensure that the certificate actually belongs to the client. Note that this is possible if the certificate has a signing capability; a pair of keys, public and private, is involved. The certificate for fixed Diffie-Hellman cannot be verified this way.

Phase IV: Finalizing and Finishing

In Phase IV, the client and server send messages to change cipher specification and to finish the handshaking protocol. Four messages are exchanged in this phase



Phase IV: Finalizing and Finishing

Change Cipher Spec The client sends a Change Cipher Spec message to show that it has moved all of the cipher suite set and the parameters from the pending state to the active state. This message is actually part of the Change Cipher Spec Protocol that we will discuss later.

Finished The next message is also sent by the client. It is a Finished message that announces the end of the handshaking protocol by the client.

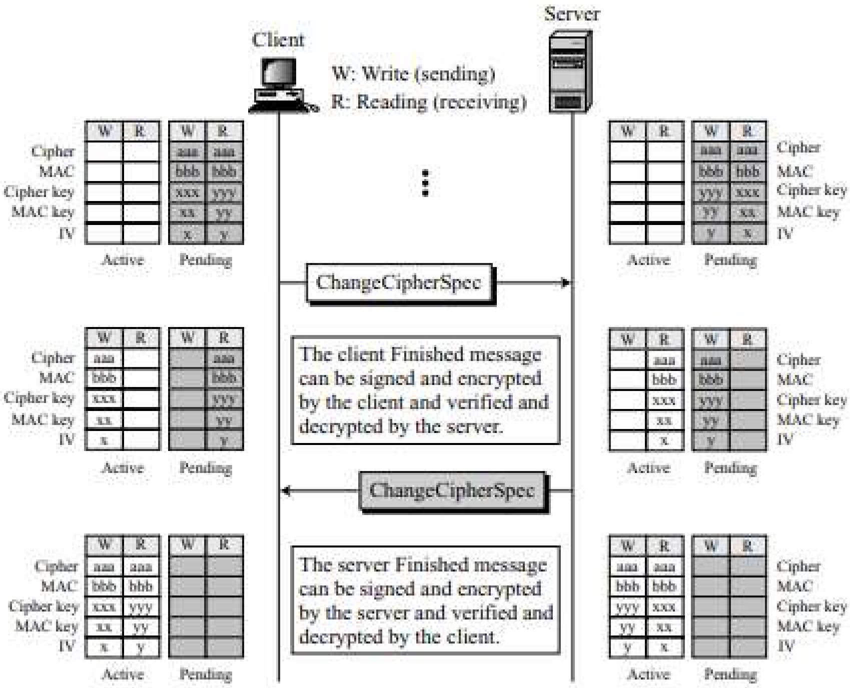
Change Cipher Spec The server sends a Change Cipher Spec message to show that it has also moved all of the cipher suite set and parameters from the pending state to the active state. This message is part of the Change Cipher Spec Protocol, which will be discussed later

Finished Finally, the server sends a Finished message to show that handshaking is totally completed.

Change Cipher Spec Protocol

We have seen that the negotiation of the cipher suite and the generation of cryptographic secrets are formed gradually during the Handshake Protocol. The question now is: When can the two parties use these parameter secrets? SSL mandates that the parties cannot use these parameters or secrets until they have sent or received a special message, the Change Cipher Spec message, which is exchanged during the Handshake protocol and defined in the Change Cipher Spec

Protocol. The reason is that the issue is not just sending or receiving a message. The sender and the receiver need two states, not one. One state, the pending state, keeps track of the parameters and secrets. The other state, the active state, holds parameters and secrets used by the Record Protocol to sign/verify or encrypt/decrypt messages. In addition, each state holds two sets of values: read (inbound) and write (outbound).



Change Cipher Spec Protocol

First the client sends a Change Cipher Spec message. After the client sends this message, it moves the write (outbound) parameters from pending to active. The client can now use these parameters to sign or encrypt outbound messages. After the receiver receives this message, it moves the read (inbound) parameters from the pending to the active state.

Now the server can verify and decrypt messages. This means that the Finished message sent by the client can be signed and encrypted by the client and verified and decrypted by the server.

The server sends the Change Cipher Spec message after receiving the Finish message from the client. After sending this message it moves the write (outbound) parameters from pending to active. The server can now use these parameters to sign or encrypt outbound messages.

After the client receives this message, it moves the read (inbound) parameters from the pending to the active state. Now the client can verify and decrypt messages. Of course, after the exchanged Finished messages, both parties can communicate in both directions using the read/write active parameters.

Alert Protocol

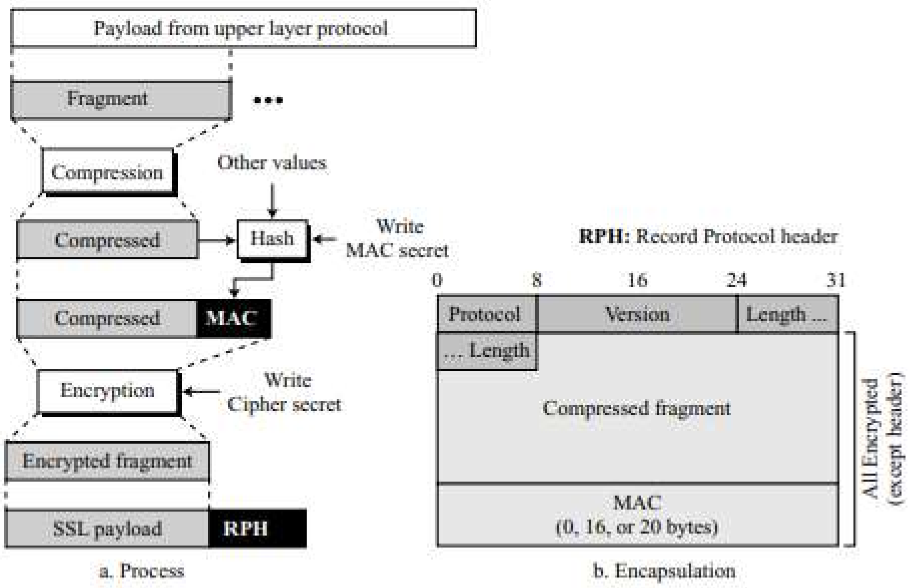
SSL uses the Alert Protocol for reporting errors and abnormal conditions. It has only one message type, the Alert message, that describes the problem and its level (warning or fatal).

Value	Description	Meaning
0	*CloseNotify*	Sender will not send any more messages.
10	*UnexpectedMessage*	An inappropriate message received.
20	*BadRecordMAC*	An incorrect MAC received.
30	*DecompressionFailure*	Unable to decompress appropriately.
40	*HandshakeFailure*	Sender unable to finalize the handshake.
41	*NoCertificate*	Client has no certificate to send.
42	*BadCertificate*	Received certificate corrupted.
43	*UnsupportedCertificate*	Type of received certificate is not supported.
44	*CertificateRevoked*	Signer has revoked the certificate.
45	*CertificateExpired*	Certificate expired.
46	*CertificateUnknown*	Certificate unknown.
47	*IllegalParameter*	An out-of-range or inconsistent field.

Alert Protocol

Record Protocol

The Record Protocol carries messages from the upper layer (Handshake Protocol, Change Cipher Spec Protocol, Alert Protocol, or application layer). The message is fragmented and optionally compressed; a MAC is added to the compressed message using the negotiated hash algorithm. The compressed fragment and the MAC are encrypted using the negotiated encryption algorithm. Finally, the SSL header is added to the encrypted message. this process at the sender. The process at the receiver is reversed.



Record Protocol

Note, however, that this process can only be done when the cryptographic parameters are in the active state. Messages sent before the movement from pending to active are neither signed nor encrypted. However, in the next sections, we will see some messages in the Handshake Protocol that use some defined hash values for message integrity.

9 a) Write a short note on security protocols and security policy in network layer.

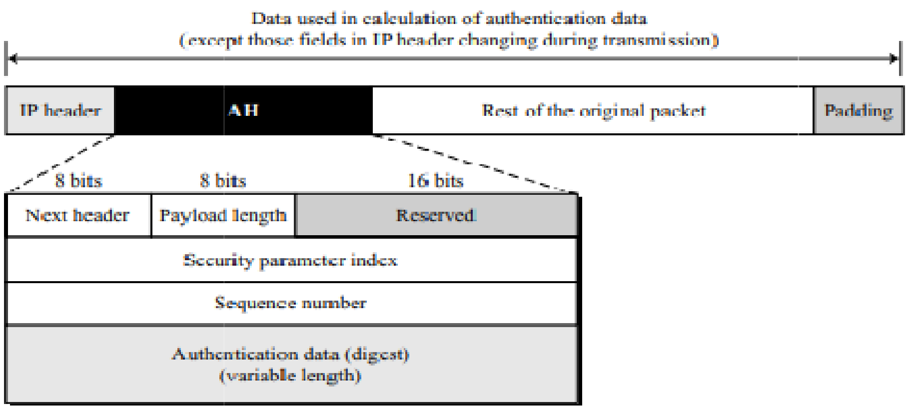
Security protocols------4M   Security policy----3M

SECURITY PROTOCOLS

IP Sec defines two protocols the Authentication Header (AH) Protocol and the Encapsulating Security Payload (ESP) Protocol to provide authentication and/or encryption for packets at the IP level.

Authentication Header (AH)

The **Authentication Header (AH)** Protocol is designed to authenticate the source host and to ensure the integrity of the payload carried in the IP packet. The protocol uses a hash function and a symmetric key to create a message digest; the digest is inserted in the authentication header. The AH is then placed in the appropriate location, based on the mode (transport or tunnel). The fields and the position of the authentication header in transport mode.



Authentication Header (AH) protocol

When an IP datagram carries an authentication header, the original value in the protocol field of the IP header is replaced by the value 51. A field inside the authentication header (the next header field) holds the original value of the protocol field (the type of payload being carried by the IP datagram). The addition of an authentication header follows these steps:

1. An authentication header is added to the payload with the authentication data field set to 0.

2. Padding may be added to make the total length even for a particular hashing algorithm.

3. Hashing is based on the total packet. However, only those fields of the IP header that do not change during transmission are included in the calculation of the message digest (authentication data).

4. The authentication data are inserted in the authentication header

5. The IP header is added after changing the value of the protocol field to 51. A brief description of each field follows:

❑ **Next header**. The 8-bit next header field defines the type of payload carried by the IP datagram (such as TCP, UDP, ICMP, or OSPF). It has the same function as the protocol field in the IP header before encapsulation. In other words, the process copies the value of the protocol field in the IP datagram to this field. The value of the protocol field in the new IP datagram is now set to 51 to show that the packet carries an authentication header.

❑ **Payload length**. The name of this 8-bit field is misleading. It does not define the length of the payload; it defines the length of the authentication header in 4-byte multiples, but it does not include the first 8 bytes.

❑ **Security parameter index**. The 32-bit security parameter index (SPI) field plays the role of a virtual circuit identifier and is the same for all packets sent during a connection called a Security Association (discussed later).

❑ **Sequence number**. A 32-bit sequence number provides ordering information for a sequence of datagrams. The sequence numbers prevent a playback. Note that the sequence number is not repeated even if a packet is retransmitted. A sequence number does not wrap around after it reaches 232; a new connection must be established.

❑ **Authentication data.**Finally, the authentication data field is the result of applying a hash function to the entire IP datagram except for the fields that are changed during transit (e.g., time-to-live).

SECURITY POLICY

Another import aspect of IP Sec is the **Security Policy (SP),** which defines the type of security applied to a packet when it is to be sent or when it has arrived. Before using the SAD, discussed in the previous section, a host must determine the predefined policy for the packet.

Security Policy Database

Each host that is using the IP Sec protocol needs to keep a **Security Policy Database (SPD).** Again, there is a need for an inbound SPD and an outbound SPD. Each entry in the SPD can be accessed using a sextuple index: source address, destination address, name, protocol, source port, and destination port

Index	Policy
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	
< SA, DA, Name, P, SPort, DPort >	

Legend:

SA: Source Address　　SPort: Source Port
DA: Destination Address　　DPort: Destination Port
P: Protocol

Source and destination addresses can be unicast, multicast, or wildcard addresses. The name usually defines a DNS entity. The protocol is either AH or ESP. The source and destination ports are the port addresses for the process running at the source and destination hosts.

Outbound SPD

When a packet is to be sent out, the outbound SPD is consulted. Figure 18.13 shows the processing of a packet by a sender.

The input to the outbound SPD is the sextuple index; the output is one of the three following cases:

1. **Drop.**This means that the packet defined by the index cannot be sent; it is dropped.

2. Bypass. This means that there is no policy for the packet with this policy index; the packet is sent, bypassing the security header application.

3. Apply. In this case, the security header is applied. Two situations may occur.

a. If an outbound SA is already established, the triple SA index is returned that selects the corresponding SA from the outbound SAD. The AH or ESP header is formed; encryption, authentication, or both are applied based on the SA selected. The packet is transmitted.

b.If an outbound SA is not established yet, the Internet Key Exchange (IKE) protocol (see the next section) is called to create an outbound and inbound SA for this traffic. The outbound SA is added to the outbound SAD by the source; the inbound SA is added to the inbound SAD by the destination.

Inbound SPD

When a packet arrives, the inbound SPD is consulted. Each entry in the inbound SPD is also accessed using the same sextuple index. Figure 18.14 shows the processing of a packet by a receiver.

The input to the inbound SPD is the sextuple index; the output is one of the three following cases:

1. Discard. This means that the packet defined by that policy must be dropped.

2. Bypass. This means that there is no policy for a packet with this policy index; the packet is processed, ignoring the information from AH or ESP header. The packet is delivered to the transport layer.

3. Apply. In this case, the security header must be processed. Two situations may occur:

a. If an inbound SA is already established, the triple SA index is returned that selects the corresponding inbound SA from the inbound SAD. Decryption, authentication, or both are applied. If the packet passes the security criteria, the AH or ESP header is discarded and the packet is delivered to the transport layer.

b.If an SA is not yet established, the packet must be discarded.

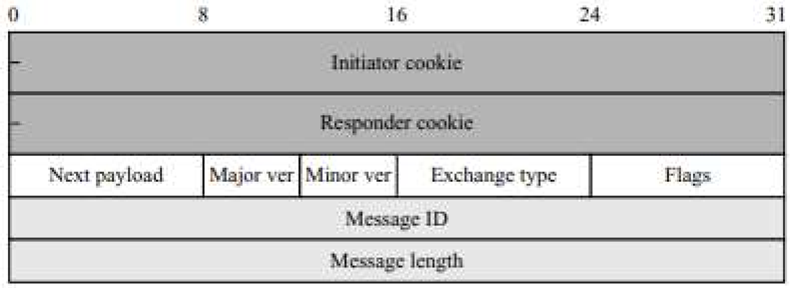
9 b) Describe any four ISAKMP payload types listing the parameters of the payload.

<u>Diaram----2M Explanation---5M</u>

ISAKMP payload

The ISAKMP protocol is designed to carry messages for the IKE exchange.

General Header



❑ **Initiator cookie**. This 32-bit field defines the cookie of the entity that initiates the SA establishment, SA notification, or SA deletion.

❑ **Responder cookie**. This 32-bit field defines the cookie of the responding entity. The value of this field is 0 when the initiator sends the first message.

❑ **Next payload**. This 8-bit field defines the type of payload that immediately follows the header. We discuss the different types of payload in the next section. Table 18.5 Encryption algorithms Value Description 1 DES 2 IDEA 3 Blowfish 4 RC5 5 3DES 6 CAST 7 AES Figure 18.29 ISAKMP general header Responder cookie Initiator cookie Message ID Message length Next payload Exchange type Flags Major version Minor version

❑ **Major version**. This 4-bit version defines the major version of the protocol. Currently, the value of this field is 1.

❑ **Minor version**. This 4-bit version defines the minor version of the protocol. Currently, the value of this field is 0.

❑ Exchange type. This 8-bit field defines the type of exchange that is being carried by the ISAKMP packets. We have discussed the different exchange types in the previous section.

❑ **Flags.** This is an 8-bit field in which each bit defines an option for the exchange. So far only the three least significant bits are defined. The encryption bit, when set to 1, specifies that the rest of the payload will be encrypted using the encryption key and the algorithm defined by SA. The commitment bit, when set to 1, specifies that encryption material is not received before the establishment of the SA. The authentication bit, when set to 1, specifies that the rest of the payload, though not encrypted, is authenticated for integrity.

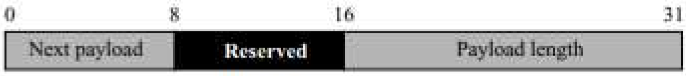
❑ **Message ID.** This 32-bit field is the unique message identity that defines the protocol state. This field is used only during the second phase of negotiation and is set to 0 during the first phase.

❑ **Message length.** Because different payloads can be added to each packet, the length of a message can be different for each packet. This 32-bit field defines the length of the total message, including the header and all payload

Payloads

The payloads are actually designed to carry messages. The types of payloads

Each payload has a generic header and some specific fields. The format of the generic header

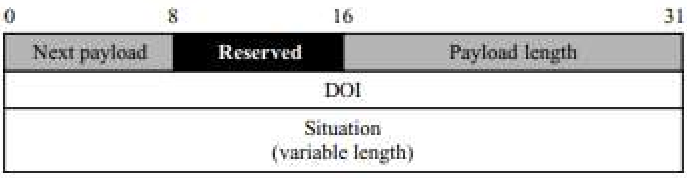


❑ **Next payload**. This 8-bit field identifies the type of the next payload. When there is no next payload, the value of this field is 0. Note that there is no type field for the current payload. The type of the current payload is determined by the previous payload or the general header (if the payload is the first one).

❑ **Payload length**. This 16-bit field defines the length of the total payload (including the generic header) in bytes.

SA Payload

The SA payload is used to negotiate security parameters. However, these parameters are not included in the SA payload; they are included in two other payloads (proposal and transform) that we will discuss later. An SA payload is followed by one or more proposal payloads, and each proposal payload is followed by one or more transform payloads. The SA payload just defines the domain of interpretation field and the situation field.



SA Payload

The fields in the generic header have been discussed. The descriptions of the other fields follow:

❑ **Domain of interpretation (DOI).** This is a 32-bit field. For phase I, a value of 0 for this field defines a generic SA; a value of 1 defines IP Sec.

❑ **Situation.** This is a variable-length field that defines the situation under which the negotiation takes place.

Proposal Payload

The proposal payload initiates the mechanism of negotiation. Although by itself it does not propose any parameters, it does define the protocol identification and the SPI. The parameters for negotiation are sent in the transform payload that follows. Each proposal payload is followed by one or more transform payloads that give alternative sets of parameters

0	8	16	24 31
Next payload	Reserved	Payload length	
Proposal #	Protocol ID	SPI size	No. of transforms
SPI (variable length)			

Proposal Payload

❑ **Proposal #.** The initiator defines a number for the proposal so that the responder can refer to it. Note that an SA payload can include several proposal payloads. If all of the proposals belong to the same set of protocols, the proposal number must be the same for each protocol in the set. Otherwise, the proposals must have different numbers.

❑ **Protocol ID**. This 8-bit field defines the protocol for the negotiation. For example, IKE phase1 = 0, ESP = 1, AH = 2, etc.

❑ **SPI size**. This 8-bit field defines the size of the SPI in bytes.

❑ **Number of Transforms**. This 8-bit field defines the number of transform payloads that will follow this proposal payload.

❑ **SPI**. This variable-length field is the actual SPI. Note that if the SPI does not fill the 32-bit space, no padding is added.

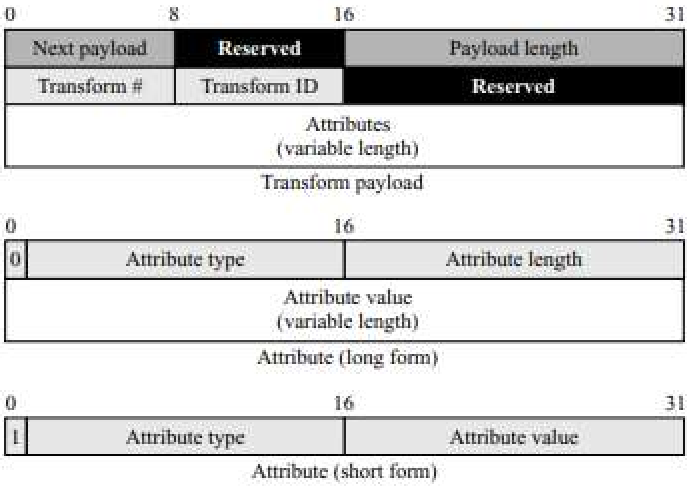
Transform Payload

The transform payload actually carries attributes of the SA negotiation. Figure 18.33 shows the format of the transform payload. The fields in the generic header have been discussed. The descriptions of the other fields follow:

❑ **Transform #.** This 8-bit field defines the transform number. If there is more than one transform payload in a proposal payload, then each must have its own number.

❑ **Transform ID**. This 8-bit field defines the identity of the payload.

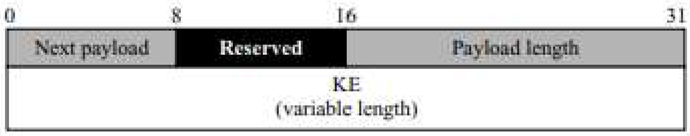
❑ **Attributes**. Each transform payload can carry several attributes. Each attribute itself can have three or two subfields (see Figure 18.33). The attribute type subfield defines the type of attribute as defined in the DOI. The attribute length subfield, if present, defines the length of the attribute value. The attribute value field is two bytes in the short form or of variable-length in the long form.



Transform Payload

Key-Exchange Payload

The key exchange payload is used in those exchanges that need to send preliminary keys that are used for creating session keys. For example, it can be used to send a Diffie-Hellman half-key. the format of the key-exchange payload.



Key-Exchange Payload

Scheme prepared by Signature of the HOD, IT Dept.

Paper Evaluators:

S.No	**Name Of the College**	**Name of the Faculty**	**Signature**