End Semester Examination (Winter) Subject: Cryptography and Network Security (CSD505)
Mark: 100 Time: 3 Hours Session: 2022-23
(Answer any **FIVE** questions)

1(a) Explain the typical attacks on encrypted messages. (4)
ANS:

Table 3.1 Types of Attacks on Encrypted Messages

Type of Attack	Known to Cryptanalyst
Ciphertext Only	■ Encryption algorithm ■ Ciphertext
Known Plaintext	■ Encryption algorithm ■ Ciphertext ■ One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	■ Encryption algorithm ■ Ciphertext ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	■ Encryption algorithm ■ Ciphertext ■ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	■ Encryption algorithm ■ Ciphertext ■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key ■ Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

(b) Write encryption and decryption algorithms for Playfair cipher. Compare it with the monoalphabetic cipher. (4+2)
ANS:

Playfair Key Matrix

- Use a *5 x 5* matrix.
- Fill in with letters of a key (without duplicates).
- Fill the rest of matrix with other remaining letters.
- Ex., let key = *playfair*

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

Encrypting and Decrypting

Plaintext is encrypted two letters at a time.

1. If a pair is a repeated letter, insert filler like 'X'.
2. If both letters fall in the same row, replace each with the letter to its right (circularly).
3. If both letters fall in the same column, replace each with the letter below it (circularly).
4. Otherwise, each letter is replaced by the letter in the same row but in the column of the other letter of the pair.

Playfair decryption is opposite of encryption as shown above.

Comparison: The security analysis of mono-alphabetic cipher is given below:

Security complexity

- Now we have a total of 26! ≈ 4 x 10^{26} keys, which is much more than in Cesar, Additive, Multiplicative and Affine ciphers.
- It is secure against brute-force attacks.
- But it is not secure against Frequency Analysis attack

Frequency Analysis

- Human languages are not random.
- In English (or any language) certain letters are used more often than others.
- In English, E is by far the most common letter, followed by T, R, N, I, O, A, S.
- If we look at a ciphertext, certain ciphertext letters are going to appear more often than others.
- It would be a good guess that the letters that occur most often in the ciphertext are actually the most common English letters as mentioned here.

Thus, not even the large number of keys in a monoalphabetic cipher provides security. One approach to improving security is to encrypt multiple letters at a time, and such approach is the **Playfair Cipher** (the best known such cipher).

It's security level is 25! (lesser than Monoalphabetic, 26!), however, it can be broken as it still leaves some structure of plaintext intact. Thus, unless the keyword is long, the last few rows of the matrix are predictable.

(c) Relate Vernam cipher to One-time pad cryptosystem and explain their security protection. (4+2)

ANS:

VERNAM CIPHER The ultimate defense against polyalphabetic and other cryptosystems is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.

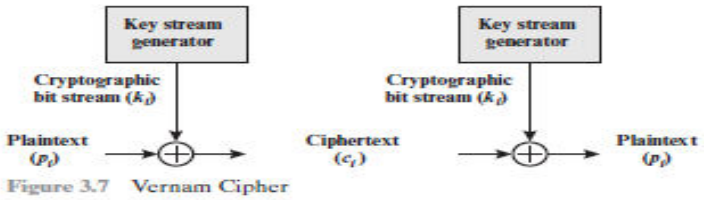


Figure 3.7 Vernam Cipher

His system works on binary data (bits) rather than letters. The system can be expressed succinctly as follows (Figure 3.7):

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

The essence of this technique is the means of construction of the key. Vernam proposed the use of a running loop of tape that eventually repeated the key, so that in fact the system worked with a very long but repeating keyword. Although such a scheme, with a long key, presents formidable cryptanalytic difficulties, it can be broken with sufficient ciphertext, the use of known or probable plaintext sequences, or both.
On the other hand, for OTP cryptosystem, an Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is unbreakable. It produces random output that bears no statistical relationship to the plaintext. Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.
The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.
2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

Because of these difficulties, the one-time pad is of limited utility and is useful primarily for low-bandwidth channels requiring very high security. The one-time pad is the only cryptosystem that exhibits what is referred to as *perfect secrecy*.

(d) Encrypt and decrypt the word '*freedom*' using Affine cipher with the key pair (*5*, *8*). (4)

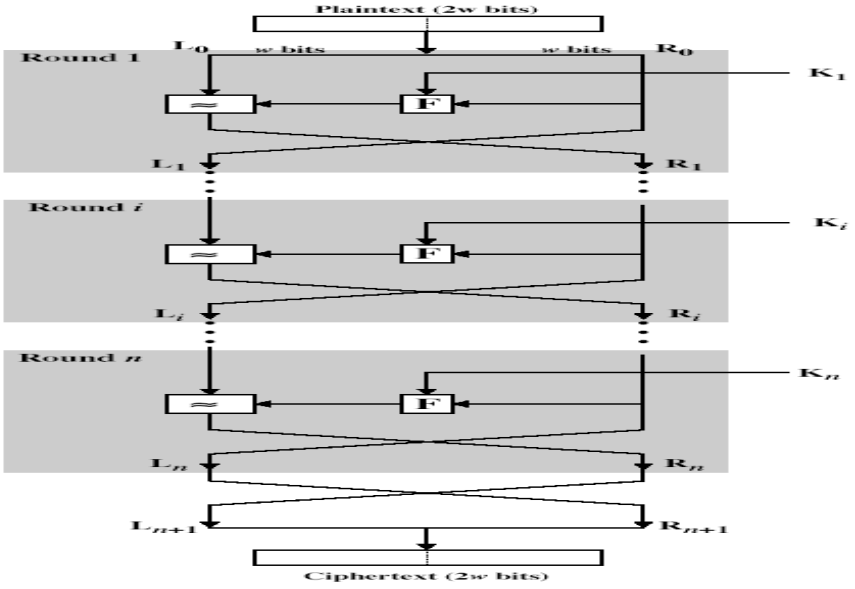
ANS:

On encryption, Ciphertext: HPCCXAQ, and on decryption, Plaintext: **freedom**, and for them, the following method is used:

c = 5p+8 mod 26 and p=21(c-8) mod 26, where c and p are ciphertext and plaintext letters, respectively.

2(a) With respect to Feistel cipher, explain the parameters and design choices involved in developing an actual cryptosystem. (4)

ANS: The Feistel cipher structure is shown below



The encryption and decryption processes are:

- $LE_i = RE_{i-1}$
- $RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$

- $RE_{i-1} = LE_i$
- $LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i)$

- **The exact realization of Feistel cipher depends on the choice of the following parameters and design parameters:**
- **Block size**
  - increasing size improves security, but reduces the speed of encryption/decryption process.
- **Key size**
  - increasing size improves security, makes exhaustive key searching harder, but may decrease the speed of encryption/decryption process.
- **Number of rounds**
  - Multiple rounds offer increasing security.
- **Subkey generation**
  - Greater complexity can make analysis harder, but slows cipher
- **Round function**
  - Greater complexity can make analysis harder, but slows cipher
- **Fast software en/decryption**
  - are more recent concerns for practical use and testing
- Easy of analysis: Although E/D algorithms are complex, there is great benefit in making algorithm easy to analyse as cryptanalytic analysis develop a higher level assurance of its strength, DES does not have easily analysed functionality.

DES follows the Feistel cipher and the parameters selected are:

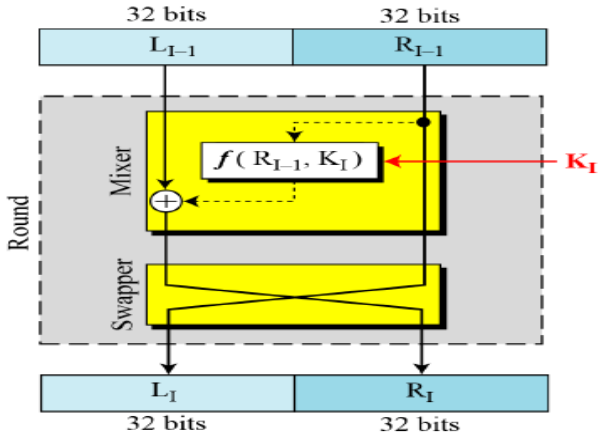
Block-size: 64 bits, Key-size: 56 bits, No. Rounds= 16, Sub-key generation: DES provides a complex method for sub-key generation that require harder to extract key and break DES; Round-function: DES uses a complex round function which is one-way both for encryption and decryption;

Fast S/W En/Decryption: DES does not support it, however, it requires in recent times for easy practical use and testing;

Easy of Analysis: DES does not have easily analysed functionality (although there is great benefit in making the algorithm easy to analysis as cryptanalytic analysis develop a higher level assurance of its strength).

(b) Draw a single round of DES with key-scheduling method, and explain the function and purposes of S-boxes.
(3+3)

ANS:



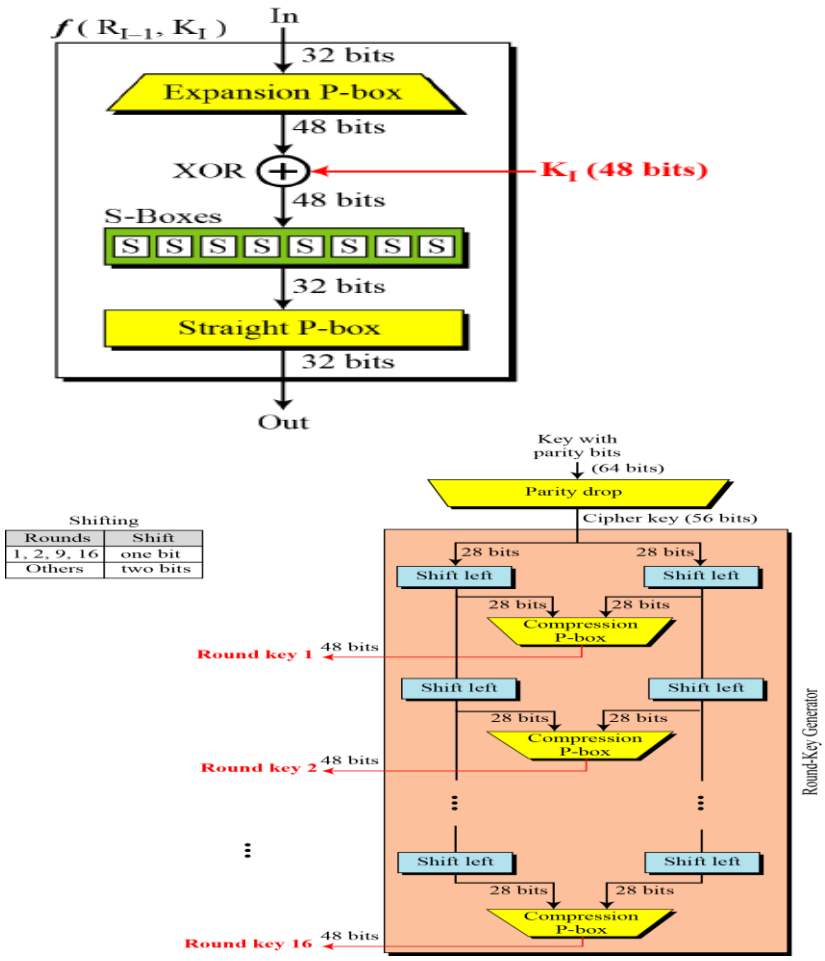
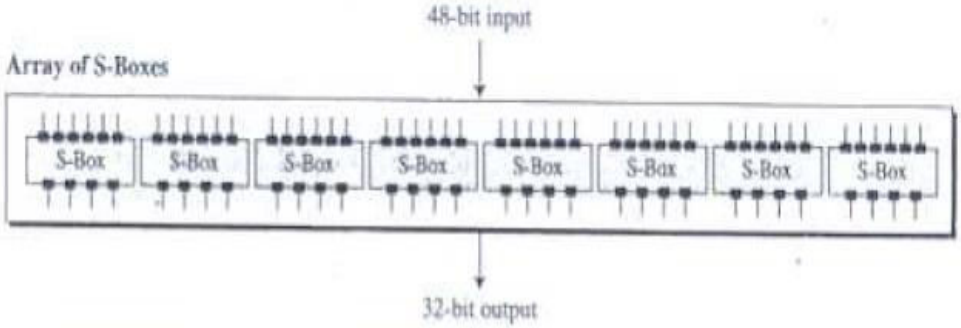


Figure 6.7 *S-boxes*



The 48-bit data from the second operation is divided into eight 6-bit chunks, and each chunk is fed into a box. The result of each box is a 4-bit chunk; when these are combined the result is a 32-bit text. The substitution in each box follows a pre-determined rule based on a 4-row by 16-column table. The combination of bits 1 and 6 of the input defines one of four rows; the combination of bits 2 through 5 defines one of the sixteen columns as shown in Figure 6.8. This will become clear in the examples.

S-boxes do the real mixing (confusion) in DES and it supports non-linearity, which is only part in DES that does it as all other parts of DES are linear. Sharp

(c) Mention the differences between AES decryption algorithm and the equivalent inverse cipher of AES. (5)

ANS:

AES decryption cipher is not identical to the encryption cipher. That is, the sequence of transformations for decryption differs from that for encryption, although the form of the key schedules for encryption and decryption is the same. This has the disadvantage that two separate software or firmware modules are needed for applications that require both encryption and decryption. There is, however, an equivalent version of the decryption algorithm that has the same structure as the encryption algorithm. The equivalent version has the same sequence of transformations as the encryption algorithm (with transformations replaced by their inverses). To achieve this equivalence, a change in key schedule is needed. The original AES E/D is shown below:

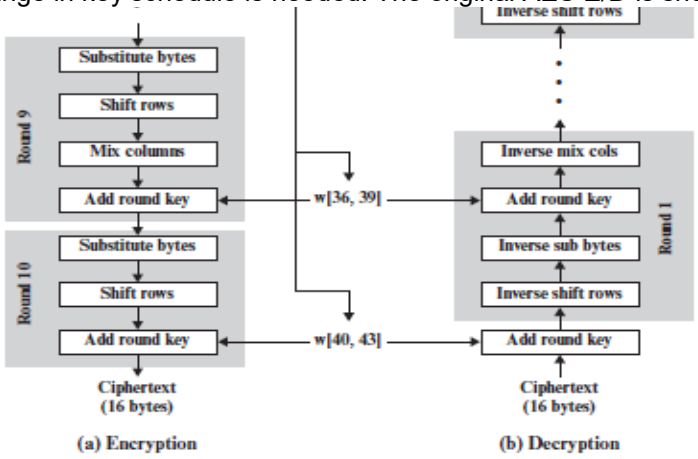


Figure 6.3 AES Encryption and Decryption

Two separate changes are needed to bring the decryption structure in line with the encryption structure. As illustrated in Figure 6.3, an encryption round has the structure SubBytes, ShiftRows, MixColumns, AddRoundKey. The standard decryption round has the structure InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Thus, the first two stages of the decryption round need to be interchanged, and the second two stages of the decryption round need to be interchanged.

Interchanging InvShiftRows and InvSubBytes InvShiftRows affects the sequence of bytes in **State** but does not alter byte contents and does not depend on byte contents to perform its transformation. InvSubBytes affects the contents of bytes in **State** but does not alter byte sequence and does not depend on byte sequence to perform its transformation. Thus, these two operations commute and can be interchanged. For a given **State** S_i,

$$\text{InvShiftRows}\,[\text{InvSubBytes}\,(S_i)] = \text{InvSubBytes}\,[\text{InvShiftRows}\,(S_i)]$$

Interchanging AddRoundKey and InvMixColumns The transformations AddRoundKey and InvMixColumns do not alter the sequence of bytes in **State**. If we view the key as a sequence of words, then both AddRoundKey and InvMixColumns operate on **State** one column at a time. These two operations are linear with respect to the column input. That is, for a given **State** S_i and a given round key w_j,

$$\text{InvMixColumns}\,(S_i \oplus w_j) = [\text{InvMixColumns}\,(S_i)] \oplus [\text{InvMixColumns}\,(w_j)]$$

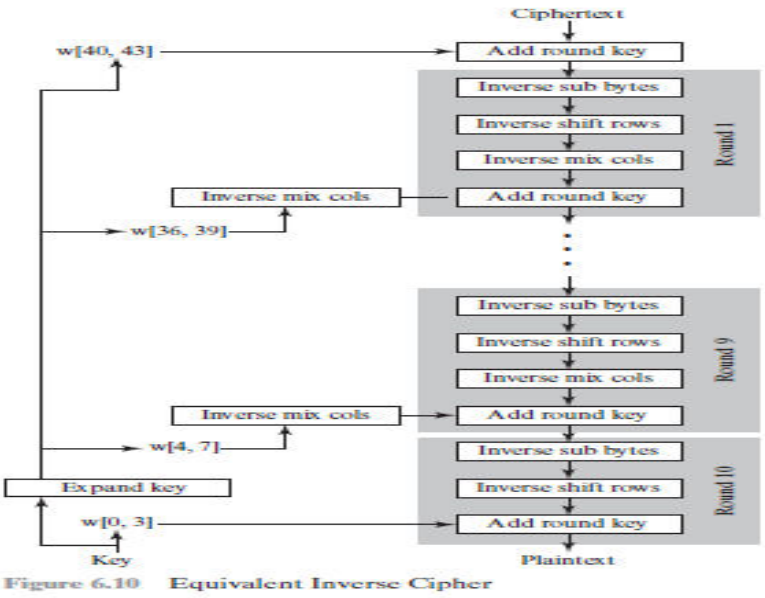


Figure 6.10 Equivalent Inverse Cipher

(d) Compute the element of AES S-box at location (row = *09*, column = *05*), where the values are hexadecimal numbers. (5)

ANS:

As an example, consider the input value {95}. The multiplicative inverse in $GF(2^8)$ is $\{95\}^{-1} = \{8A\}$, which is 10001010 in binary. Using Equation (6.2),

$$\begin{bmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} 0\\1\\0\\1\\0\\0\\0\\1 \end{bmatrix} \oplus \begin{bmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{bmatrix} = \begin{bmatrix} 1\\0\\0\\1\\0\\0\\1\\0 \end{bmatrix} \oplus \begin{bmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{bmatrix} = \begin{bmatrix} 0\\1\\0\\1\\0\\1\\0\\0 \end{bmatrix}$$

$$\begin{bmatrix} b'_0\\b'_1\\b'_2\\b'_3\\b'_4\\b'_5\\b'_6\\b'_7 \end{bmatrix} = \begin{bmatrix} 1&0&0&0&1&1&1&1\\ 1&1&0&0&0&1&1&1\\ 1&1&1&0&0&0&1&1\\ 1&1&1&1&0&0&0&1\\ 1&1&1&1&1&0&0&0\\ 0&1&1&1&1&1&0&0\\ 0&0&1&1&1&1&1&0\\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0\\b_1\\b_2\\b_3\\b_4\\b_5\\b_6\\b_7 \end{bmatrix} + \begin{bmatrix} 1\\1\\0\\0\\0\\1\\1\\0 \end{bmatrix} \qquad (6.2)$$

Answer is S-Box[95] = 2A as shown.

3(a) Write RSA algorithm. Mention its hard problem with justification, and explain proper choices of the private and public keys. (2+2+3)

ANS:

Bob runs RSA algorithm (say)

- Bob chooses two distinct large primes p, and q (of size ≥ *512* bits) and computes n = pq.
- Bob calculates ϕ(n) = (p-1)(q-1) and chooses e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- Bob computes d with $de \equiv 1 \bmod \phi(n)$.
- Bob's public key: (e, n) and private key : (d)
- Bob makes n and e public, and keeps d secret.

Alice encrypts m as $c \equiv m^e \bmod n$ and sends c to Bob.

Bob decrypts by computing $m \equiv c^d \bmod n$

Plaintext: $M < n$

Factorization Attack: The security of RSA is based on the idea that the modulus is so large that it is not feasible to factor it in a reasonable time. The attacker attempts to factor n. If this can be done then it is a simple way to compute $\phi(n)$ and subsequently d will be derived.

THE FACTORING PROBLEM We can identify three approaches to attacking RSA mathematically.

1. Factor n into its two prime factors. This enables calculation of $\phi(n) = (p - 1) \times (q - 1)$, which in turn enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
2. Determine $\phi(n)$ directly, without first determining p and q. Again, this enables determination of $d \equiv e^{-1} \pmod{\phi(n)}$.
3. Determine d directly, without first determining $\phi(n)$.

To reduce the encryption time, it is tempting to use a small encryption exponent e.

The common value is considered as e =3 (11, 17 & like these) or with less no. of 1's in binary representation of e as 10001=17

Bob may think that using low decryption exponent would make the decryption process faster.

Wiener showed that if $d < \frac{1}{3}n^{1/4}$ and $q < p < 2q$ then the attacker can factor n in polynomial time.

In RSA, the recommendation is to have $d \geq \frac{1}{3}n^{1/4}$ to prevent low decryption exponent attack.

(b) Describe the known plaintext attack on ElGamal cryptosystem and provide its protection with validation. (2)

ANS:

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key	
Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2 K^{-1}) \bmod q$

If k is used for more than one block, knowledge of one block M_1 of the message enables the user to compute other blocks as follows. Let

$$C_{1,1} = \alpha^k \bmod q; C_{2,1} = KM_1 \bmod q$$
$$C_{1,2} = \alpha^k \bmod q; C_{2,2} = KM_2 \bmod q$$

Then,

$$\frac{C_{2,1}}{C_{2,2}} = \frac{KM_1 \bmod q}{KM_2 \bmod q} = \frac{M_1 \bmod q}{M_2 \bmod q}$$

If M_1 is known, then M_2 is easily computed as

$$M_2 = (C_{2,1})^{-1} C_{2,2} M_1 \bmod q$$

(c) Write algorithms for the non-superincreasing and superincreasing knapsack problems and compute their time complexities. (3+3)

ANS: The knapsack problem is as follows:

$a = [a_1, a_2, \ldots, a_n]$ and $x = [x_1, x_2, \ldots, x_n]$ where $x_i = 0$ or 1.

$$s = knapsackSum\ (a, x) = x_1a_1 + x_2a_2 + \cdots + x_na_n$$

a and *x* are two *n*-tuple, 1st is predefined set. X defines which of a are to be dropped.

- Given a and x, it is easy to calculate s. However, given s and a it is difficult to find x.
- That is, s = knapsackSum(a, x) is easy, but x=inv_knapsackSum(s, a) is difficult.
- Since there are n items, there are 2^n possible combinations.
- We go through all combinations and find the one with total weight equal to s, then, Complexity will be $O(2^n)$

Intro

$$a_i \geq a_1 + a_2 + \ldots + a_{i-1}$$

Algorithm *knapsacksum and inv_knapsackSum for a superincreasing k-tuple*

```
knapsackSum (x [1 ... k], a [1 ... k])
{
    s ← 0
    for (i = 1 to k)
    {
      s ← s + a_i × x_i
    }
    return s
}
```

```
inv_knapsackSum (s, a [1 ... k])
{
    for (i = k down to 1)
    {
        if s ≥ a_i
        {
            x_i ← 1
            s ← s − a_i
        }
        else x_i ← 0
    }
    return x [1 ... k]
}
```

Complexity of inv_knapsackSum: O(n)

(d) Let the ciphertext $c \equiv 100\ (mod\ 4661 = 59 \times 79)$ obtained using Rabin cipher. Decrypt c to get the message m, where $c \equiv m^2\ (mod\ 4661)$. (5)

ANS: Since 4661= 59×70, the

factorization of his secret key, which is $4661 = 59 \cdot 79$. Let $p = 59$ and $q = 79$ in this case. Now, he starts to compute the following

$$m_p = 100^{(59+1)/4} = 100^{60/4} = 100^{15} \equiv 49 \pmod{59}$$

$$m_q = 100^{(79+1)/4} = 100^{80/4} = 100^{20} \equiv 10 \pmod{79}.$$

Then finding y_p and y_q using extended Euclidean algorithm:

$$\begin{aligned} 79 &= 59 - 20 \\ 59 &= 20 \cdot 2 + 19 \\ 20 &= 19 + 1 \\ 1 &= 20 - 19 \\ &= 20 - 59 + 20 \cdot 2 \\ &= 59 - 20 \cdot 3 \\ &= 59 - 79 \cdot 3 + 59 \cdot 3 \\ &= 59 \cdot 4 + 79 \cdot (-3) \end{aligned}$$

so $y_p = 4$ and $y_q = -3$.

Then, using CRT

$$\begin{aligned} m_1 &= -4 \cdot 59 \cdot 10 + 3 \cdot 79 \cdot 10 &\equiv\ & 4592 \pmod{4661} \\ m_2 &= 4661 - 4692 &\equiv\ & 69 \pmod{4661} \\ m_3 &= -4 \cdot 59 \cdot 10 - 3 \cdot 79 \cdot 10 &\equiv\ & 10 \pmod{4661} \\ m_4 &= 4661 - 10 &\equiv\ & 4651 \pmod{4661}. \end{aligned}$$

Now Bob have to use extra information that is sent by Alice in order to select which one is her favorite number. There is no way that Bob would know it is either 10, 69, 4592, 4651 that is Alice's original message since all of them squared give exactly 100 modulo 4661. In that case, further instruction or information is needed.

4(a) Compare the security strengths of MAC function with a symmetric encryption technique in terms of known (*message, tag*) and (*plaintext, ciphertext*) pairs, respectively. (5)

ANS:

A MAC, also known as a cryptographic checksum, is generated by a function C of the form

$$T = \text{MAC}(K, M)$$

where M is a variable-length message, K is a secret key shared only by sender and receiver, and $\text{MAC}(K, M)$ is the fixed-length authenticator, sometimes called a **tag**. The tag is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the tag.

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key. In particular, for a ciphertext-only attack, the opponent, given ciphertext C, performs $P_i = \text{D}(K_i, C)$ for all possible key values K_i until a P_i is produced that matches the form of acceptable plaintext.

In the case of a MAC, the considerations are entirely different. In general, the MAC function is a many-to-one function, due to the many-to-one nature of the function. Using brute-force methods, how would an opponent attempt to discover a key? If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose $k > n$; that is, suppose that the key size is greater than the MAC size. Then, given a known M_1 and T_1, with $T_1 = \text{MAC}(K, M_1)$, the cryptanalyst can perform $T_i = \text{MAC}(K_i, M_1)$ for all possible key values k_i. At least one key is guaranteed to produce a match of $T_i = T_1$. Note that a total of 2^k tags will be produced, but there are only $2^n < 2^k$ different tag values. Thus, a number of keys will produce the correct tag and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack.

- **Round 1**

  Given: $M_1, T_1 = \text{MAC}(K, M_1)$
  Compute $T_i = \text{MAC}(K_i, M_1)$ for all 2^k keys
  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

  Given: $M_2, T_2 = \text{MAC}(K, M_2)$
  Compute $T_i = \text{MAC}(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1
  Number of matches $\approx 2^{(k-2\times n)}$

And so on. On average, α rounds will be needed $k = \alpha \times n$. For example, if an 80-bit key is used and the tag is 32 bits, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only a single key, which must be the one used by the sender.

Thus, in general, the security of MAC is much more than any encryption technique.

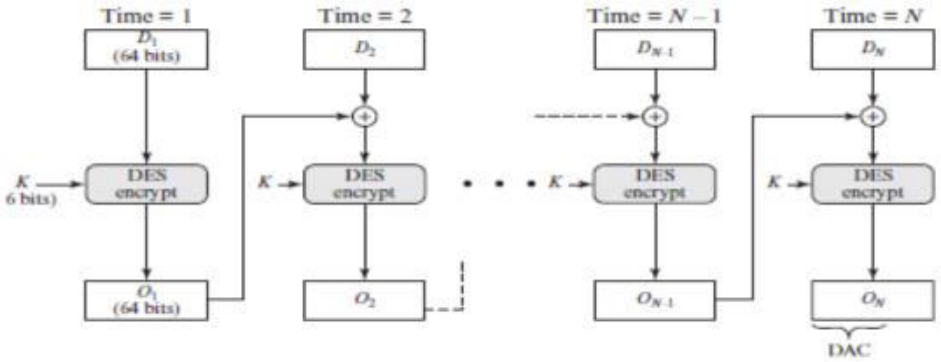
(b) Develop a method of designing MAC function using DES encryption technique. (5)

ANS:

Data Authentication Algorithm (DAA), based on DES, has been one of the most widely used MACs for a number of years. The algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17). However, as we discuss subsequently, security weaknesses in this algorithm have been discovered, and it is being replaced by newer and stronger algorithms.

The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure 6.4) with an initialization vector of zero. The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks: $D_1, D_2, \ldots, D_N$. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm E and a secret key K, a data authentication code (DAC) is calculated as follows (Figure 12.7).

$$
\begin{aligned}
O_1 &= \mathrm{E}(K, D) \\
O_2 &= \mathrm{E}(K, [D_2 \oplus O_1]) \\
O_3 &= \mathrm{E}(K, [D_3 \oplus O_2]) \\
&\cdot \\
&\cdot \\
&\cdot \\
O_N &= \mathrm{E}(K, [D_N \oplus O_{N-1}])
\end{aligned}
$$



(c) Describe the signing and verification procedures of the ElGamal signature scheme. (5)

ANS:

Alice's public key: (p, α, β) where $\beta \equiv \alpha^a \bmod p$ and private key: a

In order for Alice to sign a message m, she does the following:

1. Selects a secret random k such that $\gcd(k, p-1) = 1$
2. Computes $r \equiv \alpha^k \pmod p$
3. Computes $s \equiv k^{-1}(m - ar) \pmod{p-1}$

The signed message is the triple (m, r, s).

Bob can verify the signature as follows:

1. Download Alice's public key (p, α, β).
2. Compute $v_1 \equiv \beta^r r^s \pmod p$, and $v_2 \equiv \alpha^m \pmod p$.
3. The signature is declared valid if and only if $v_1 \equiv v_2 \pmod p$.

We now show that the verification procedure works. Assume the signature is valid. Since $s \equiv k^{-1}(m - ar) \pmod{p-1}$, we have $sk \equiv m - ar \pmod{p-1}$, so $m \equiv sk + ar \pmod{p-1}$. Therefore (recall that a congruence mod $p-1$ in the exponent yields an overall congruence mod p),

$$v_2 \equiv \alpha^m \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r (\alpha^k)^s \equiv \beta^r r^s \equiv v_1 \pmod p.$$

(d) Compute the probability of at least *2*, out of *23* random people, have the same birthday, and relate the consequence to a cryptographic hash function. (5)

ANS:

Problem Statement: How many people are needed at a party such that there is a reasonable chance (probability more than 0.5) at least two people have the same birthday?

So

$$P(\text{no collision among 2 people}) = \left(1 - \frac{1}{365}\right)$$

If a third person joins the party, he or she can collide with both of the people already there, hence:

$$P(\text{no collision among 3 people}) = \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right)$$

Consequently, the probability for t people having no birthday collision is given by:

$$P(\text{no collision among } t \text{ people}) = \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdots \left(1 - \frac{t-1}{365}\right)$$

For $t = 366$ people we will have a collision with probability 1 since a year has only 365 days. We return now to our initial question: how many people are needed to have a 50% chance of two colliding birthdays? Surprisingly—following from the equations above—it only requires 23 people to obtain a probability of about 0.5 for a birthday collision since:

$$\begin{aligned} P(\text{at least one collision}) &= 1 - P(\text{no collision}) \\ &= 1 - \left(1 - \frac{1}{365}\right) \cdots \left(1 - \frac{23-1}{365}\right) \\ &= 0.507 \approx 50\%. \end{aligned}$$

73 people with more than 99%

Note that for 40 people the probability is about 90%. Due to the surprising outcome it is often referred to as the birthday paradox.

5(a) Define and specify the uses of SAD and SPD in IPSec. (3+3)

ANS:

A Security Association can be very complex. This is particularly true if Alice wants to send messages to many people and Bob needs to receive messages from many people. In addition, each site needs to have both inbound and outbound SAs to allow bidirectional communication. In other words, we need a set of SAs that can be collected into a database. This database is called the **Security Association Database (SAD).** The database can be thought of as a two-dimensional table with each row defining a single SA. Normally, there are two SADs, one inbound and one outbound. Figure 18.11 shows the concept of outbound and inbound SADs for one entity.

< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							
< SPI, DA, P >							

Security Association Database

Legend:

SPI: Security Parameter Index
DA: Destination Address
AH/ESP: Information for either one
P: Protocol
Mode: IPSec Mode Flag
SN: Sequence Number
OF: Overflow Flag
ARW: Anti-Replay Window
LT: Lifetime
MTU: Path MTU (Maximum Transfer Unit)

When a host needs to send a packet that must carry an IPSec header, the host needs to find the corresponding entry in the outbound SAD to find the information for applying security to the packet. Similarly, when a host receives a packet that

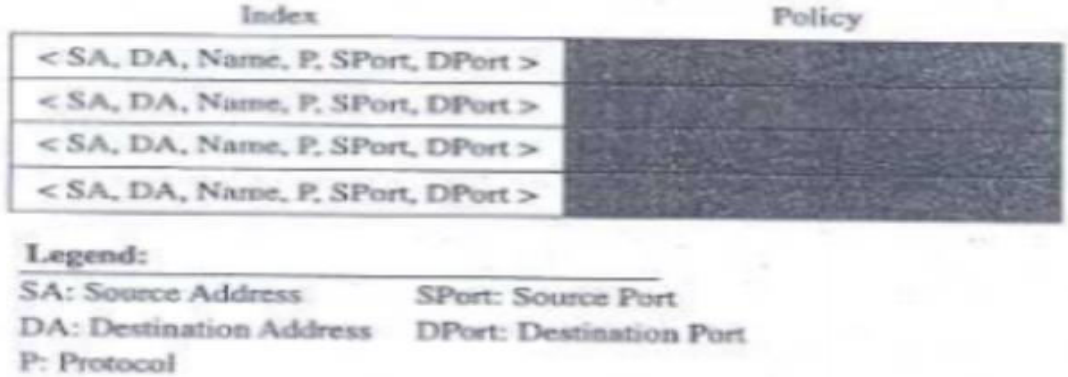
carries an IPSec header, the host needs to find the corresponding entry in the inbound SAD to find the information for checking the security of the packet. This searching must be specific in the sense that the receiving host needs to be sure that correct information is used for processing the packet. Each entry in an inbound SAD is selected using a triple index: security parameter index, destination address, and protocol.

- **Security Parameter Index.** The security parameter index (SPI) is a 32-bit number that defines the SA at the destination. As we will see later, the SPI is determined during the SA negotiation. The same SPI is included in all IPSec packets belonging to the same inbound SA.
- **Destination Address.** The second index is the destination address of the host. We need to remember that a host in the Internet normally has one unicast destination address, but it may have several multicast addresses. IPSec requires that the SAs be unique for each destination address.
- **Protocol.** IPSec has two different security protocols: AH and ESP. To separate the parameters and information used for each protocol, IPSec requires that a destination define a different SA for each protocol.

Another import aspect of IPSec 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 IPSec 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, as shown in Figure 18.12.



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.

(b) Describe the original public-key method for main-mode of phase-I of IKE protocol. (5)

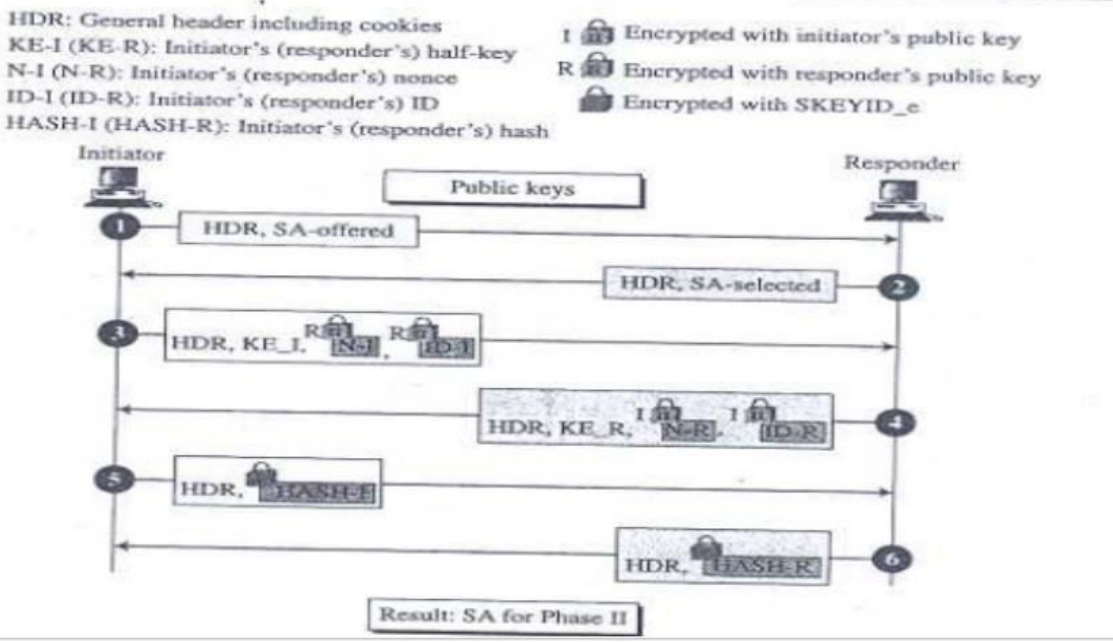
ANS:

In the **main mode,** the initiator and the responder exchange six messages. In the first two messages, they exchange cookies (to protect against a clogging attack) and negotiate the SA parameters. The initiator sends a series of proposals; the responder selects one of them. When the first two messages are exchanged, the initiator and the responder know the SA parameters and are confident that the other party exists (no clogging attack occurs).

In the third and fourth messages, the initiator and responder usually exchange their half-keys (g^i and g^r of the Diffie-Hellman method) and their nonces (for replay protection). In some methods other information is exchanged; that will be discussed later. Note that the half-keys and nonces are not sent with the first two messages because the two parties must first ensure that a clogging attack is not possible.

In the original public-key method, the initiator and the responder prove their identities by showing that they possess a private key related to their announced public key. Figure 18.21 shows the exchange of messages using the original public-key method.

Figure 18.21 *Main mode, original public-key method*



After exchanging the third and fourth messages, each party can calculate the common secret between them in addition to its individual hash digest. The common secret SKEYID (secret key ID) is dependent on the calculation method as shown below. In the equations, *prf* (pseudorandom function) is a keyed-hash function defined during the negotiation phase.

$$SKEYID = prf(\text{preshared-key}, N\text{-}I \mid N\text{-}R) \quad \text{(preshared-key method)}$$

$$SKEYID = prf(N\text{-}I \mid N\text{-}R, g^{ir}) \quad \text{(public-key method)}$$

$$SKEYID = prf(\text{hash}(N\text{-}I \mid N\text{-}R), \text{Cookie-}I \mid \text{Cookie-}R) \quad \text{(digital signature)}$$

Other common secrets are calculated as follows:

$$SKEYID\_d = prf(SKEYID, g^{ir} \mid \text{Cookie-}I \mid \text{Cookie-}R \mid 0)$$

$$SKEYID\_a = prf(SKEYID, SKEYID\_d \mid g^{ir} \mid \text{Cookie-}I \mid \text{Cookie-}R \mid 1)$$

$$SKEYID\_e = prf(SKEYID, SKEYID\_a \mid g^{ir} \mid \text{Cookie-}I \mid \text{Cookie-}R \mid 2)$$

The two parties also calculate two hash digests, HASH-I and HASH-R, which are used in three of the four methods in the main mode. The calculation is shown below:

$$HASH\text{-}I = prf(SKEYID, KE\text{-}I \mid KE\text{-}R \mid \text{Cookie-}I \mid \text{Cookie-}R \mid SA\text{-}I \mid ID\text{-}I)$$

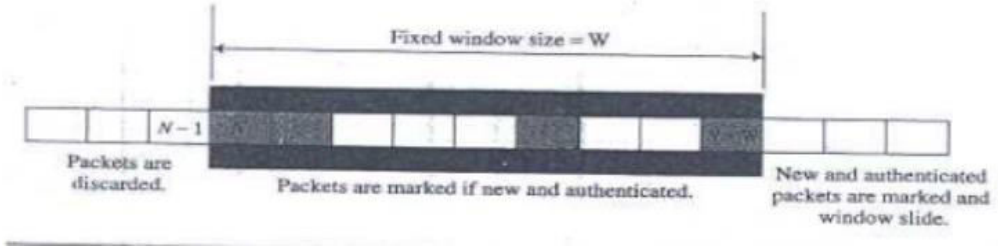
$$HASH\text{-}R = prf(SKEYID, KE\text{-}I \mid KE\text{-}R \mid \text{Cookie-}I \mid \text{Cookie-}R \mid SA\text{-}I \mid ID\text{-}R)$$

(c) How is the replay attack protected in IPSec? Explain with necessary diagrams. (5)

ANS:

In both protocols, the replay attack is prevented by using sequence numbers and a sliding receiver window. Each IPSec header contains a unique sequence number when the Security Association is established. The number starts from 0 and increases until the value reaches $2^{32} - 1$ (the size of the sequence number field is 32 bits). When the sequence number reaches the maximum, it is reset to 0 and, at the same time, the old Security Association (see the next section) is deleted and a new one is established. To prevent processing duplicate packets, IPSec mandates the use of a fixed-size window at the receiver. The size of the window is determined by the receiver with a default value of 64. Figure 18.9 shows a replay window. The window is of a fixed size, W. The shaded packets signify received packets that have been checked and authenticated.

Figure 18.9 *Replay window*



When a packet arrives at the receiver, one of three things can happen, depending on the value of the sequence number.

1. The sequence number of the packet is less than N. This puts the packet to the left of the window. In this case, the packet is discarded. It is either a duplicate or its arrival time has expired.
2. The sequence number of the packet is between N and $(N + W - 1)$, inclusive. This puts the packet inside the window. In this case, if the packet is new (not marked) and it passes the authentication test, the sequence number is marked and the packet is accepted. Otherwise, it is discarded.
3. The sequence number of the packet is greater than $(N + W - 1)$. This puts the packet to the right of the window. In this case, if the packet is authenticated, the corresponding sequence number is marked and the window slides to the right to cover the newly marked sequence number. Otherwise, the packet is discarded. Note that it may happen that a packet arrives with a sequence number much larger than $(N + W)$ (very far from the right edge of the window). In this case, the sliding of the window may cause many unmarked numbers to fall to the left of the window. These packets, when they arrive, will never be accepted; their time has expired. For example, in Figure 18.9, if a packet arrives with sequence number $(N + W + 3)$, the window slides and the left edge will be at the beginning of $(N + 3)$. This means the sequence number $(N + 2)$ is now out of the window. If a packet arrives with this sequence number, it will be discarded.

(d) Compute the Diffie-Hellman common secret-key between two remote users A and B using the following information: (4)

Prime number $p = 17$, Primitive-root $g = 3$, User A's private key $a = 15$, and User B's private key $b = 13$

ANS: A's public key = $3^{15}(mod\ 17) = 6$ and B's public key = $3^{13}(mod\ 17) = 12$, which are exchanged between each publicly.

On receiving B's public-key, A computes the common secret key $K = (12)^{15}(mod\ 17) = 10$ and similarly, B computes the same secret key as $K = (6)^{13}(mod\ 17) = 10$

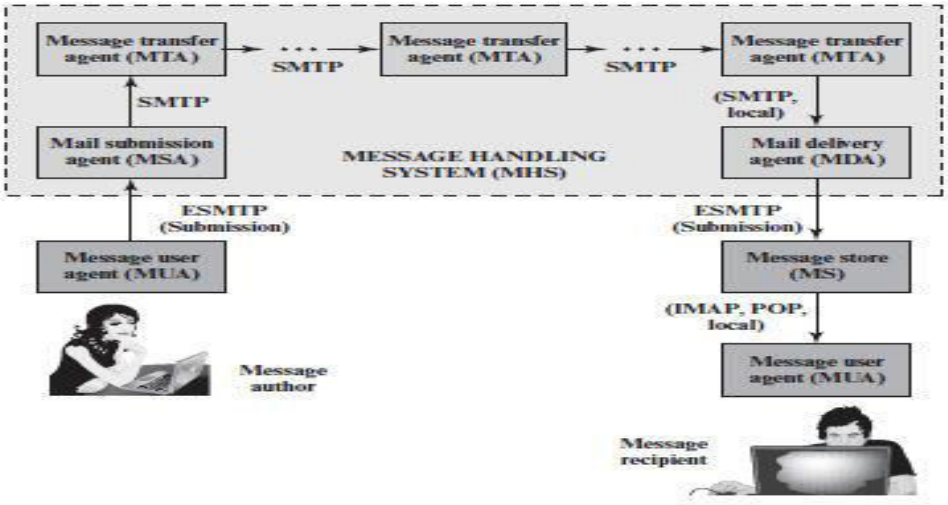
6(a) Explain the Internet email architecture with the main key components. (4)

ANS:

Internet mail architecture consists of a user world in the form of **Message User Agents (MUA),** and the transfer world, in the form of the **Message Handling Service (MHS)** (Composed of Message Transfer Agents (MTA)). The MHS accepts a message from one user and delivers it to one or more other users, creating a virtual MUA-to-MUA exchange environment. The architecture involves three types of interoperability:

- One is directly between users- messages must be formatted by the MUA on behalf of the message author so that the message can be displayed to the message recipient by the destination MUA.
- There are also interoperability requirements between the MUA and the MHS— first when a message is posted from an MUA to the MHS and later when it is delivered from the MHS to the destination MUA.
- Interoperability is required among the MTA components along the transfer path through the MHS.

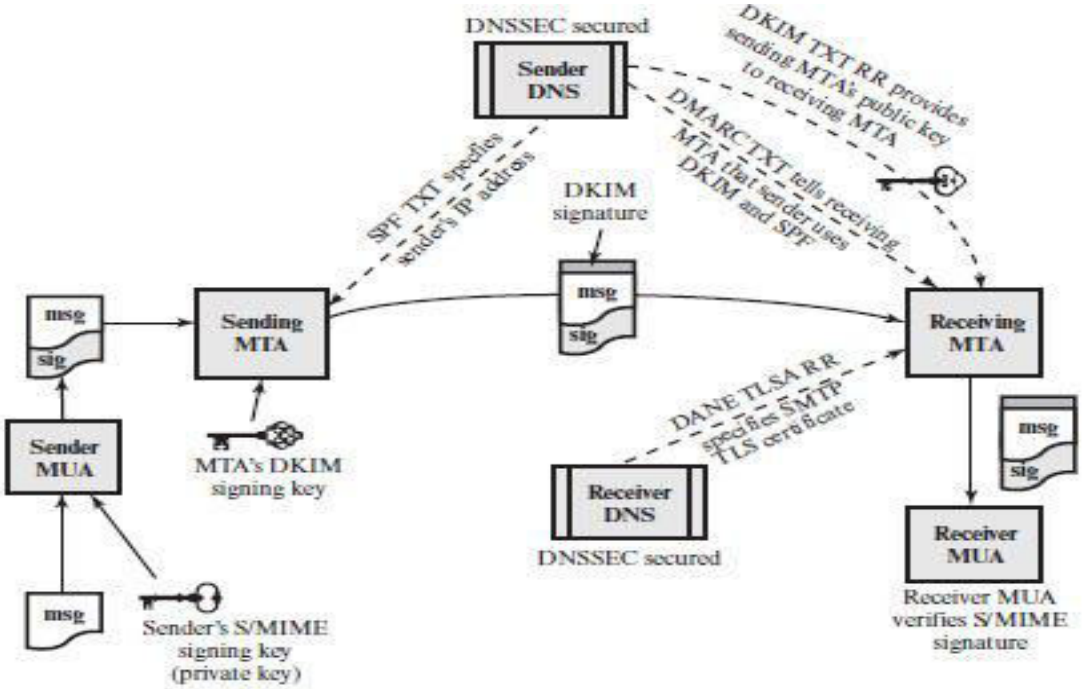
Figure below illustrates the key components of the Internet mail architecture:



- **Message User Agent (MUA):** Operates on behalf of user actors and user applications. It is their representative within the email service. Typically, this function is housed in the user's computer and is referred to as client email program or a local network email server. The author MUA formats a message and performs initial submission into the MHS via a MSA. The recipient MUA processes received mail for storage and/or display to the recipient user.
- **Mail Submission Agent (MSA):** Accepts the message submitted by an MUA and enforces the policies of the hosting domain and the requirements of Internet standards. This function may be located together with the MUA or as a separate functional model. In the latter case, the Simple Mail Transfer Protocol (SMTP) is used between the MUA and the MSA.
- **Message Transfer Agent (MTA):** Relays mail for one application-level hop. It is like a packet switch or IP router in that its job is to make routing assessments and to move the message closer to the recipients. Relaying is performed by a sequence of MTAs until the message reaches a destination MDA. An MTA also adds trace information to the message header. SMTP is used between MTAs and between an MTA and an MSA or MDA.
- ■ **Mail Delivery Agent (MDA):** Responsible for transferring the message from the MHS to the MS.
- ■ **Message Store (MS):** An MUA can employ a long-term MS. An MS can be located on a remote server or on the same machine as the MUA. Typically, an MUA retrieves messages from a remote server using POP (Post Office Protocol) or IMAP (Internet Message Access Protocol).
- Two other concepts are
  - An **administrative management domain (ADMD)** is an Internet email provider. Examples include a department that operates a local mail relay (MTA), an IT department that operates an enterprise mail relay, and an ISP that operates a public shared email service. be quite different.
  - **Domain Name System (DNS)** is a directory lookup service that provides a mapping between the name of a host on the Internet and its numerical address.

(b) Describe S/MIME based message authentication and integrity with necessary protocols and diagrams. (6)

ANS: The necessary diagram is shown below:



DANE = DNS-based Authentication of Named Entities
DKIM = DomainKeys Identified Mail
DMARC = Domain-based Message Authentication, Reporting, and Conformance
DNSSEC = Domain Name System Security Extensions
SPF = Sender Policy Framework
S/MIME = Secure Multi-Purpose Internet Mail Extensions
TLSA RR = Transport Layer Security Authentication Resource Record

SP 800-177 recommends use of a variety of standardized protocols as a means for countering these threats. These include:

- **STARTTLS:** An SMTP security extension that provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) for the entire SMTP message by running SMTP over TLS (Transport Layer Security)
- **S/MIME:** Provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) of the message body carried in SMTP messages.
- **DNS Security Extensions (DNSSEC):** Provides authentication and integrity protection of DNS data, and is an underlying tool used by various email security protocols.
- **DNS-based Authentication of Named Entities (DANE):** Is designed to overcome problems in the certificate authority (CA) system by providing an alternative channel for authenticating public keys based on DNSSEC (Domain Name System Security Extensions), with the result that the same trust relationships used to certify IP addresses are used to certify servers operating on those addresses.
- **Sender Policy Framework (SPF):** Uses the Domain Name System (DNS) to allow domain owners to create records that associate the domain name with a specific IP address range of authorized message senders. It is a simple matter for receivers to check the SPF TXT record in the DNS to confirm that the purported sender of a message is permitted to use that source address and reject mail that does not come from an authorized IP address.
- **Domain Keys Identified Mail (DKIM):** Enables an MTA to sign selected headers and the body of a message. This validates the source domain of the mail and provides message body integrity.
- **Domain-based Message Authentication, Reporting, and Conformance (DMARC):** Lets senders know the proportionate effectiveness of their SPF and DKIM policies, and signals to receivers what action should be taken in various individual and bulk attack scenarios.

(c) Write about the general PGP message format, and explain the transmission and reception of PGP messages briefly. (4+3+3)

ANS: The general PGP message format is given below:

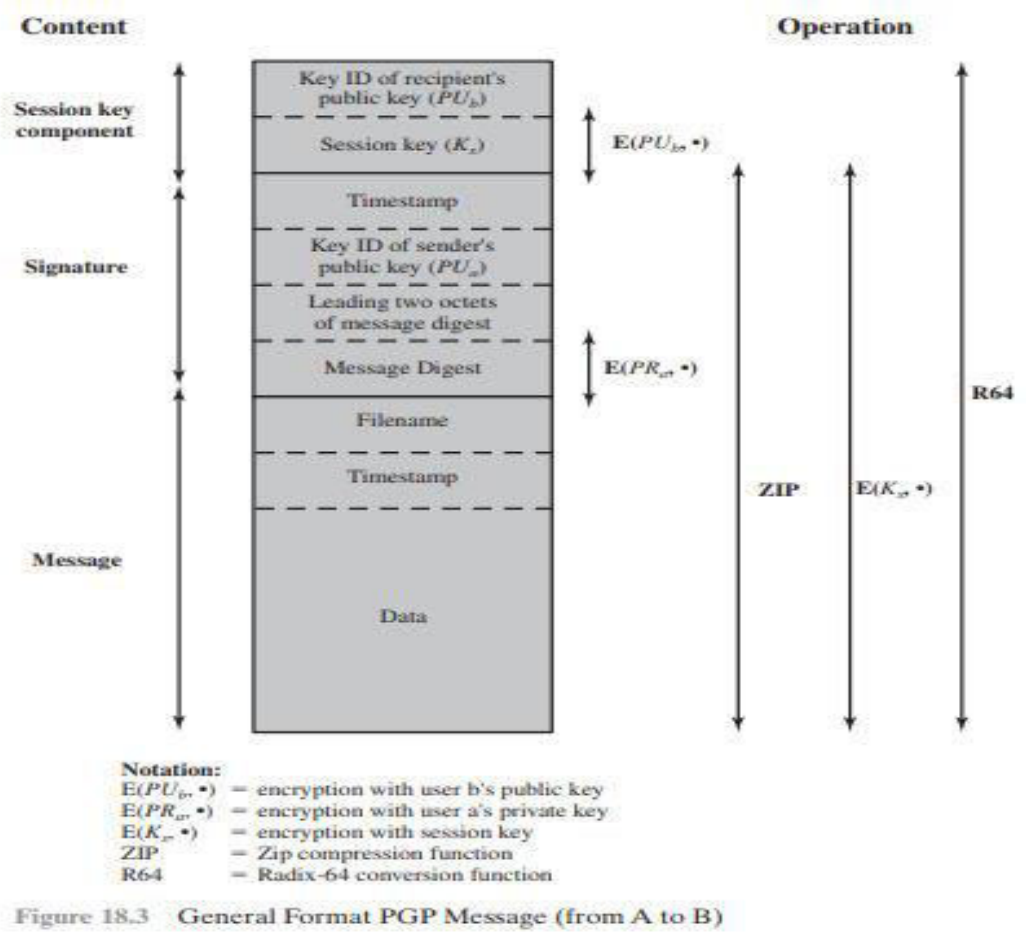


Figure 18.3 General Format PGP Message (from A to B)

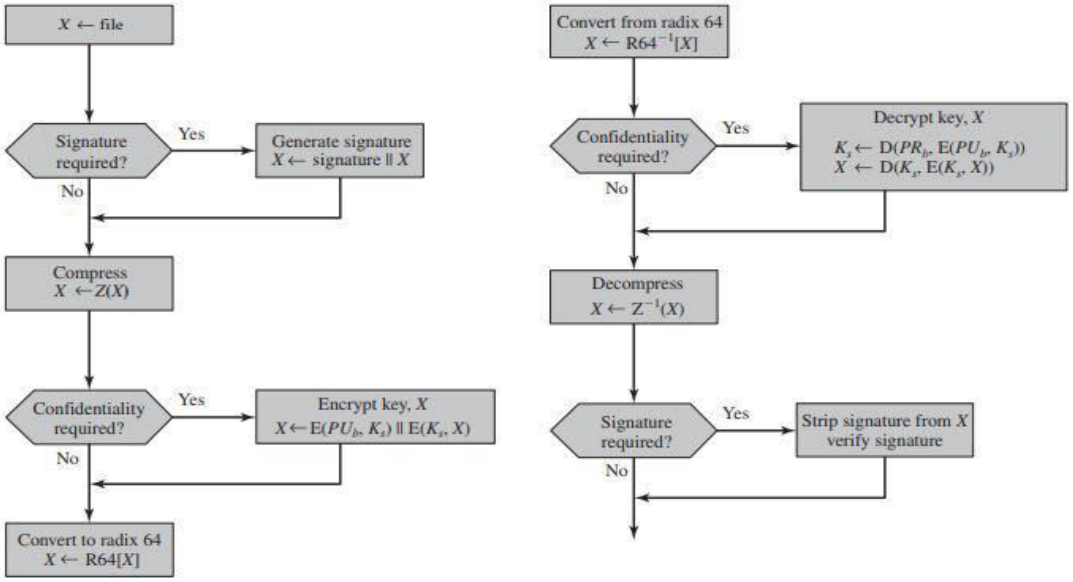
It consists of the following components:

- Message component, a signature (optional), and a session key component (optional). The message component includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The signature component includes the following.
- Timestamp: The time at which the signature was made.
- Message digest: The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest insures against replay types of attacks. The exclusion of the filename and timestamp portions of the message component ensures that detached signatures are exactly the same as attached signaturesprefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.
- Leading two octets of message digest: Enables the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.
- Key ID of sender's public key: Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest. The message component and optional signature component may be compressed using ZIP and may be encrypted using a session key.

Figures shown below represent the relationship among the four services as discussed:

• On transmission (if it is required), a signature is generated using a hash code of the uncompressed plaintext. Then the plaintext (plus signature if present) is compressed. Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the public-key encrypted symmetric encryption key. Finally, the entire block is converted to radix-64 format.

• On reception, the incoming block is first converted back from radix-64 format to binary. Then, if the message is encrypted, the recipient recovers the session key and decrypts the message. The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure 18.2 Transmission and Reception of PGP Messages