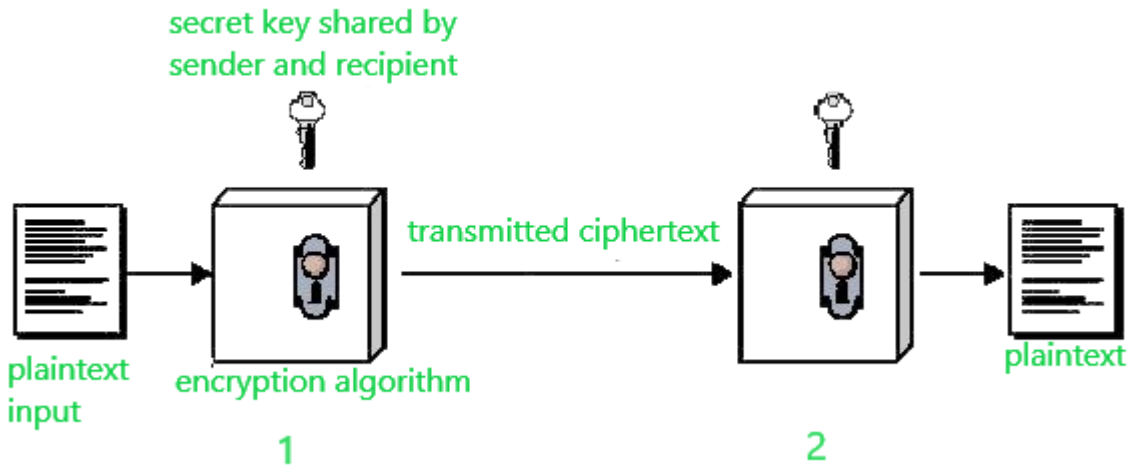
L. J Institutes of Engineering and Technology

Remedial MSE Solution INS

SEM: 7
Subject Name: Information Security
Subject Code: 3170720

1. **Explain conventional security model used for information security.**



- An original message is known as the plaintext, while the coded message is called the ciphertext.
- The process of converting from plaintext to ciphertext is known as enciphering or encryption; restoring the plaintext from the ciphertext is deciphering or decryption.
- Ciphertext is the scrambled message produced as output.
- It depends on the plaintext and the secret key.
- The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- The secret key is also input to the encryption algorithm.
- The key is a value independent of the plaintext and of the algorithm.
- The algorithm will produce a different output depending on the specific key being used at the time.
- Decryption algorithm is essentially the encryption algorithm run in reverse.
- It takes the ciphertext and the secret key and produces the original plaintext.

2. **Define the terms threat and attack. List and briefly define categories of security attacks**

- Threat: Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.
- Security Attack: Any action that compromises the security of information owned by an organization

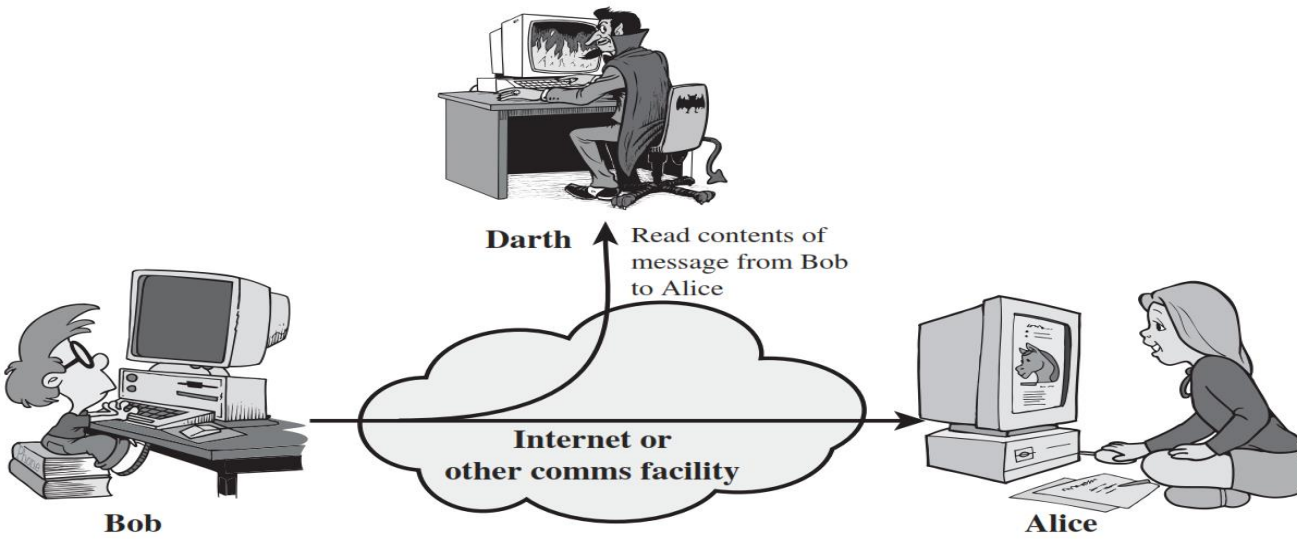
✓ A passive attack attempts to learn or make use of information from the system but does not affect system resources.
   1. Release of message contents
   2. Traffic analysis

✓ An active attack attempts to alter system resources or affect their operation.
   1. Masquerade

2. Replay
3. Modification of messages
4. Denial of service.

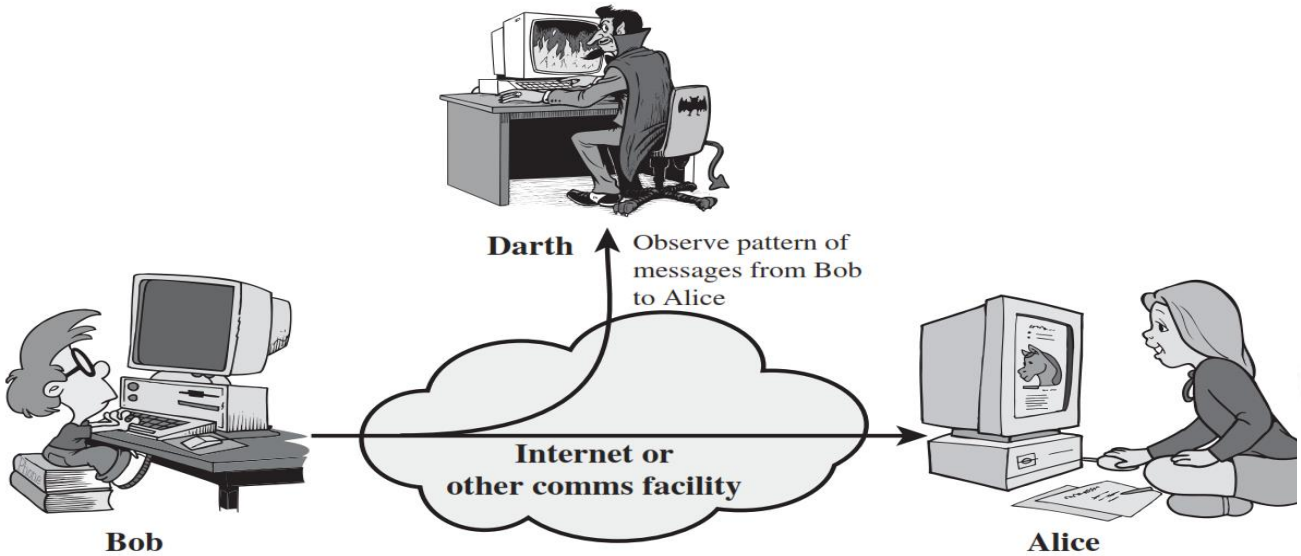
RELEASE OF MESSAGE CONTENT:

- A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information.
- Attack on Confidentiality.



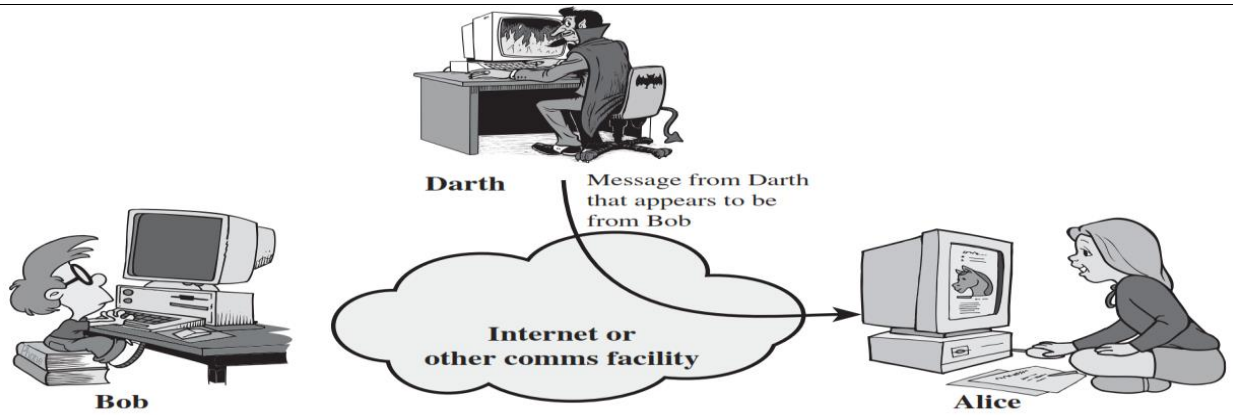
TRAFFIC ANALYSIS:

- After applying encryption attacker will observe that message and pattern of message.
- After some time attacker will be able to understand that message on base of observation.



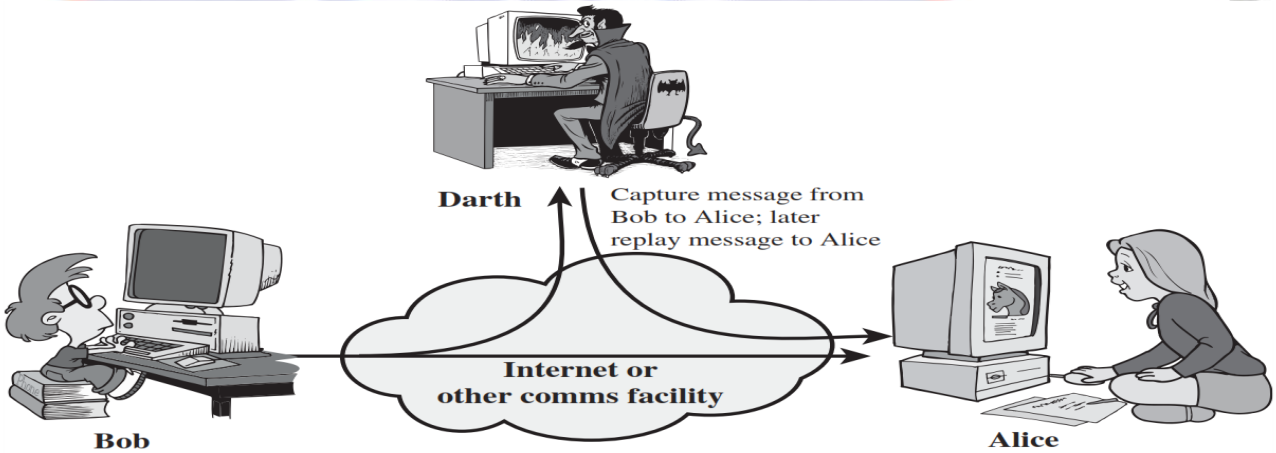
MASQUERADE:

- A masquerade takes place when one entity pretends to be a different entity.
- A masquerade attack is an attack that uses a fake identity to gain unauthorized access to personal information.
- Attack on Authentication.



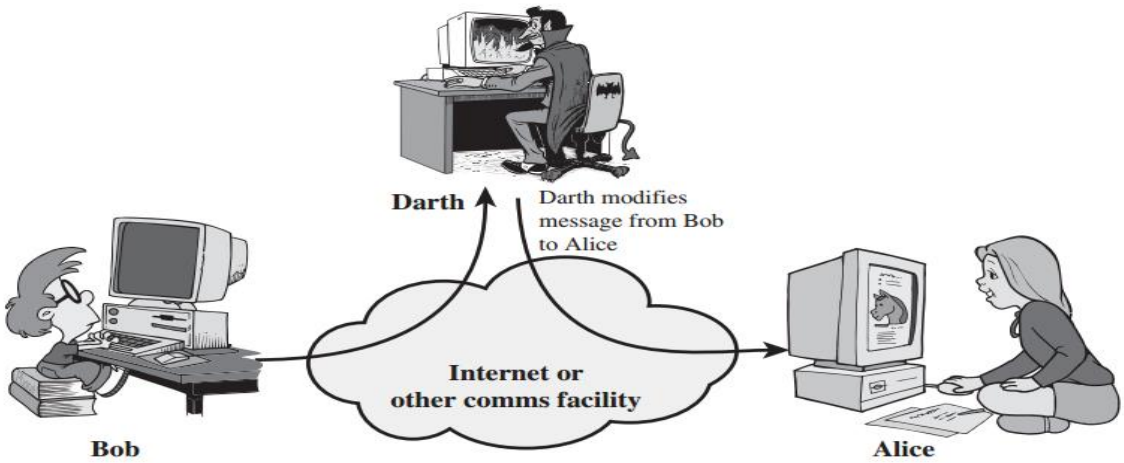
REPLAY:

- Replay attack involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- Replay attack is to replay the message sent to a network by an attacker, which was earlier sent by an authorized user.



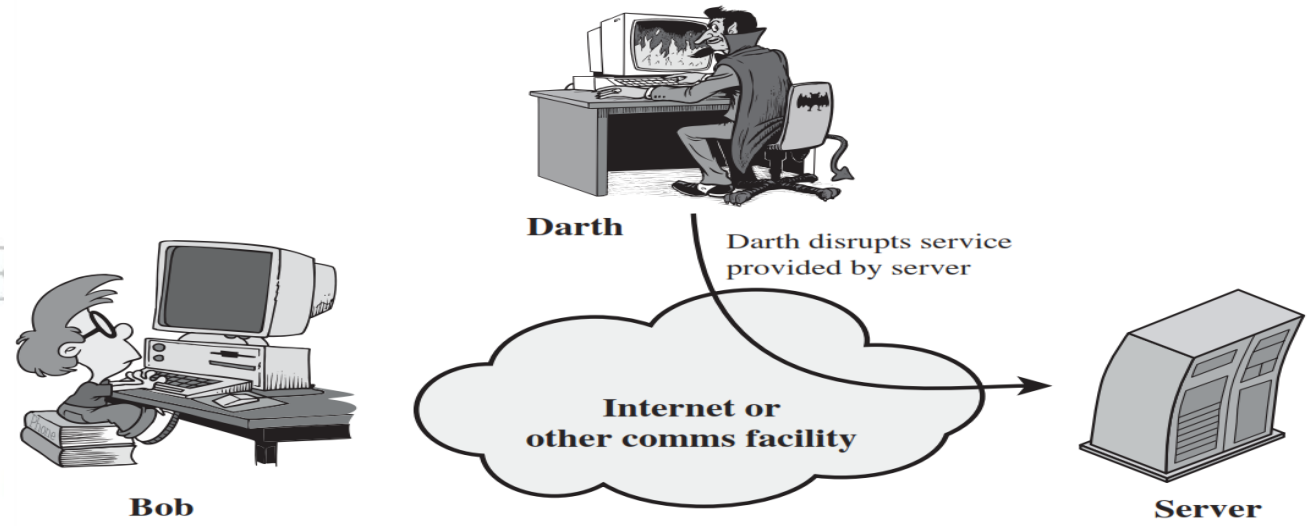
MODIFICATION OF MESSAGE:

- Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- Attack on Integrity.



DENIAL OF SERVICE:

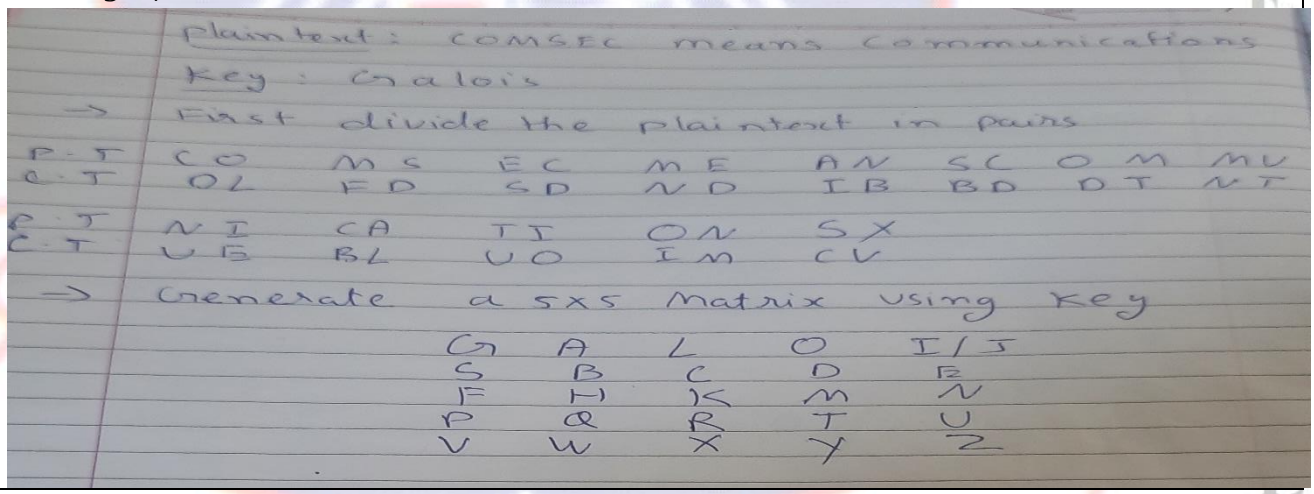
- Denial of service attack prevents the normal use or management of communications facilities.
- Sending large number of packets to block the server. Attack on Availability.

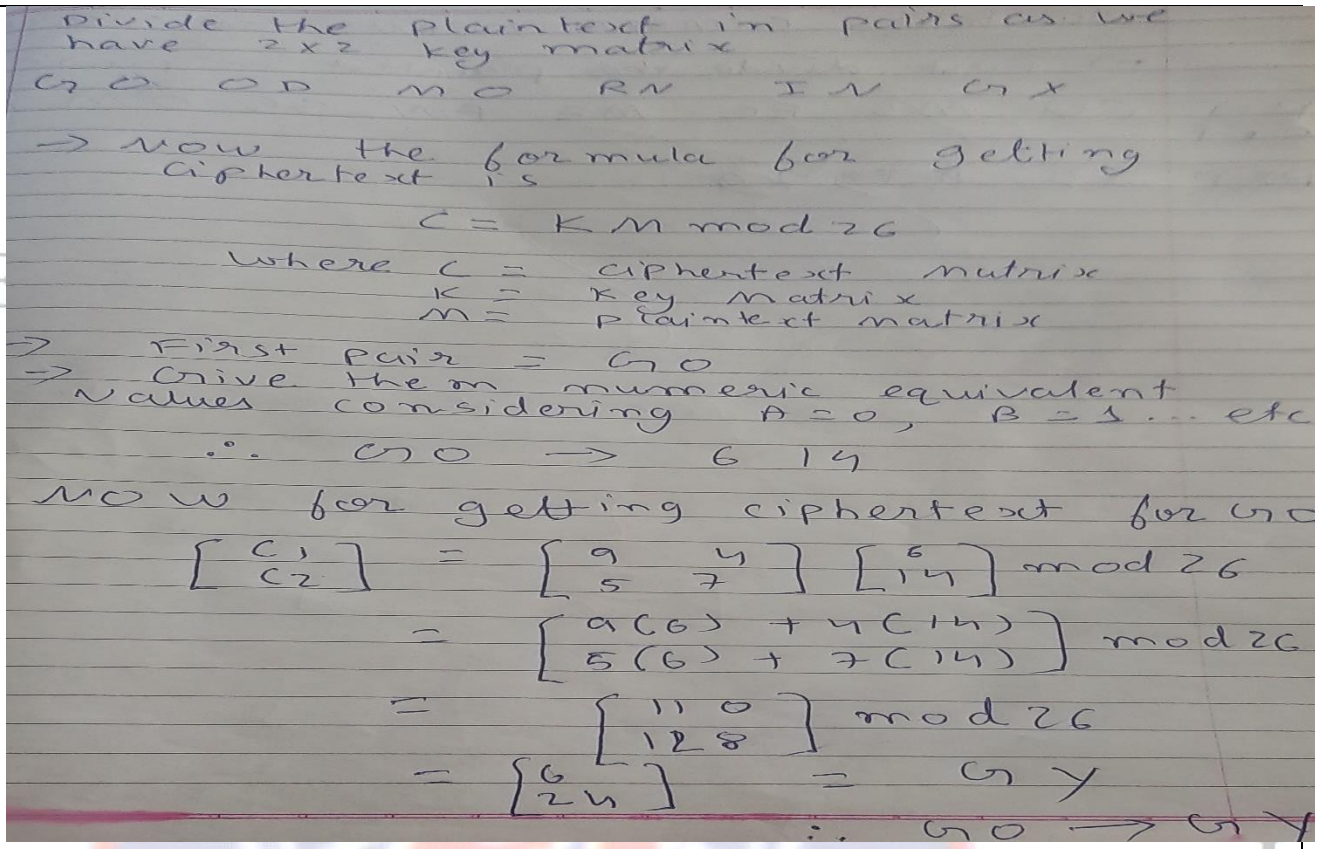


3. **(A) Encrypt the following message using playfair cipher. Message: COMSEC means communications securityKeyword: Galois.**

9 4

5 7





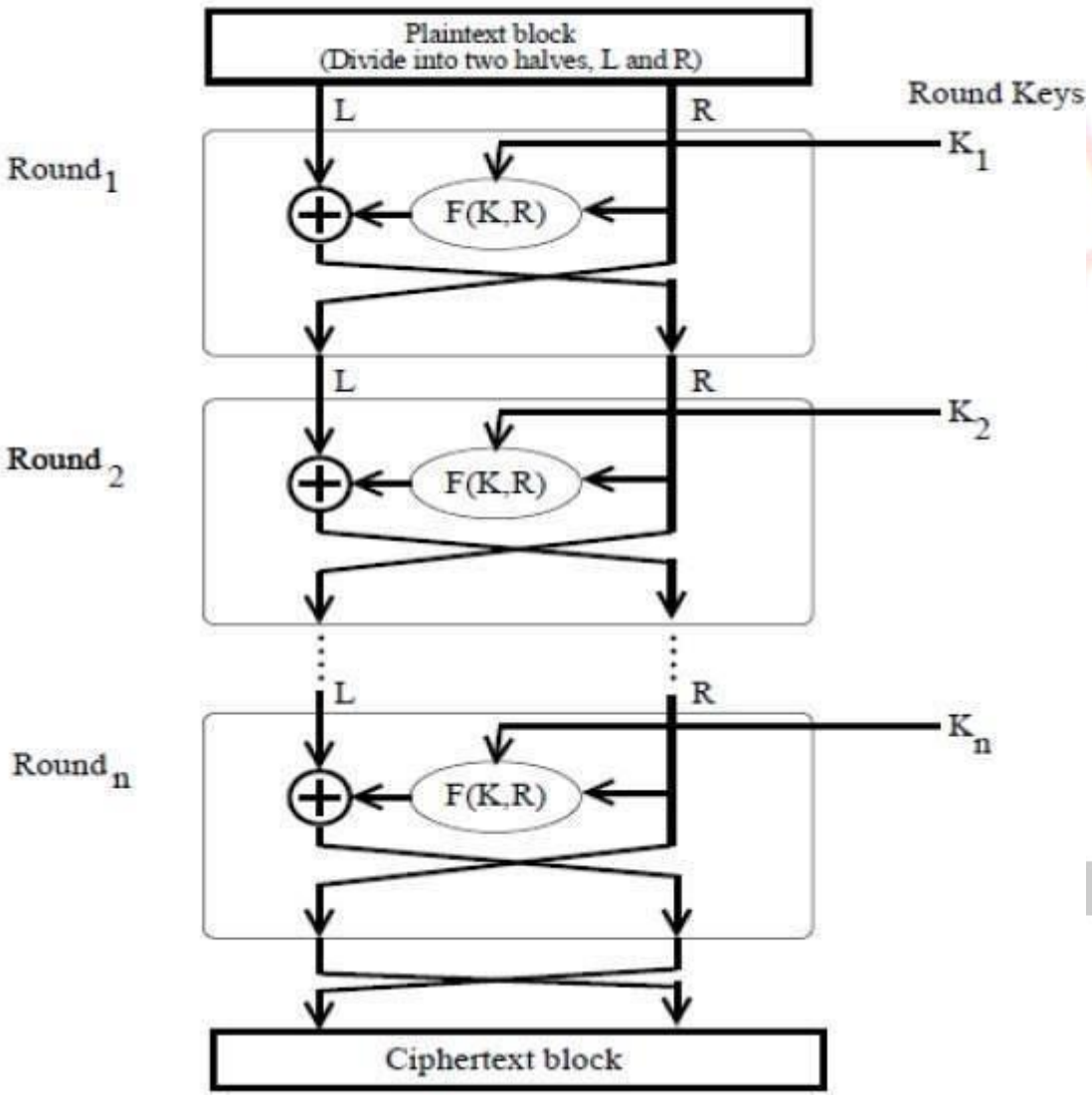
Note: In exam follow same method and solve for all other letters.

4. **(A) Explain the difference between diffusion and confusion.**
(B) What is the difference between a block cipher and a stream cipher?

BASIS FOR COMPARISON	CONFUSION	DIFFUSION
Basic	Utilized to generate vague cipher texts.	Utilized to generate obscure, plain texts.
Seeks to	Make a relation between statistics of the ciphertext and the value of the encryption key as complicated as possible.	The statistical relationship between the plaintext and ciphertext is made as complicated as possible.
Achieved through	Substitution algorithm	Transposition algorithm
Used by	Stream cipher and block cipher	Block cipher only.
Result in	Increased vagueness	Increased redundancy

Block Cipher	Stream Cipher
Processing or encoding of the plain text is done as a fixed length block one by one. A block for example could be 64 or 128 bits in size.	Processing or encoding of plain text is done bit by bit. The block size here is simply one bit.
The same key is used to encrypt each of the blocks	A different key is used to encrypt each of the bits.
A Pad added to short length blocks	Bits are processed one by one in as in a chain
Uses Symmetric Encryption and is NOT used in asymmetric encryption	High speed and low hardware complexity
Confusion factor: The key to the cipher text relationship could be really very complicated.	Key is often combined with an initialization vector
Diffusion Factor: output depends on the input in a very complex method.	Long period with no repetition
Most block ciphers are based on Feistel cipher in structure	Statistically random
Looks more like an extremely large substitution and Using the idea of a product cipher	Depends on a large key and Large liner complexity
More secure in most cases	Equally secure if properly designed
Usually more complex and slower in operation	Usually very simple and much faster
Examples of Block Cipher are: Lucifer / DES,IDEA, RC5, Blowfish etc.	Examples of Stream Cipher are: FISH, RC4, ISAAC, SEAL, SNOW etc.

5. **Explain Feistel Cipher Structure with respect to its design features**



- Feistel Cipher model is a structure or a design used to develop many block ciphers such as DES. Feistel cipher may have invertible, non-invertible and self invertible components in its design. Same encryption as well as decryption algorithm is used. A separate key is used for each round. However same round keys are used for encryption as well as decryption.

The design features of Feistel cipher that are considered while implementing any block cipher are as follow:

Block Size

- The block cipher is considered more secure if the block size is larger. But the larger block size can reduce the execution speed of encryption and decryption. Generally, the block size of a block cipher is of 64-bit. But, the modern-day block cipher such as AES has 128-bit block size.

Key Size

- The security of block cipher increases with the increasing key size. But the large key size may decrease the speed of encryption and decryption. Earlier the key of 64-bit was considered to adequate. But the modern cipher uses a key of size 128-bit.

Number of rounds

- The number of rounds also increases the security of the block cipher. More are the number of rounds more complex is the cipher.

Subkey generation function

- More the subkey generation function is complex, difficult it is for a cryptanalyst to crack it.

Round Function

- Complex round function enhances the security of the block cipher.

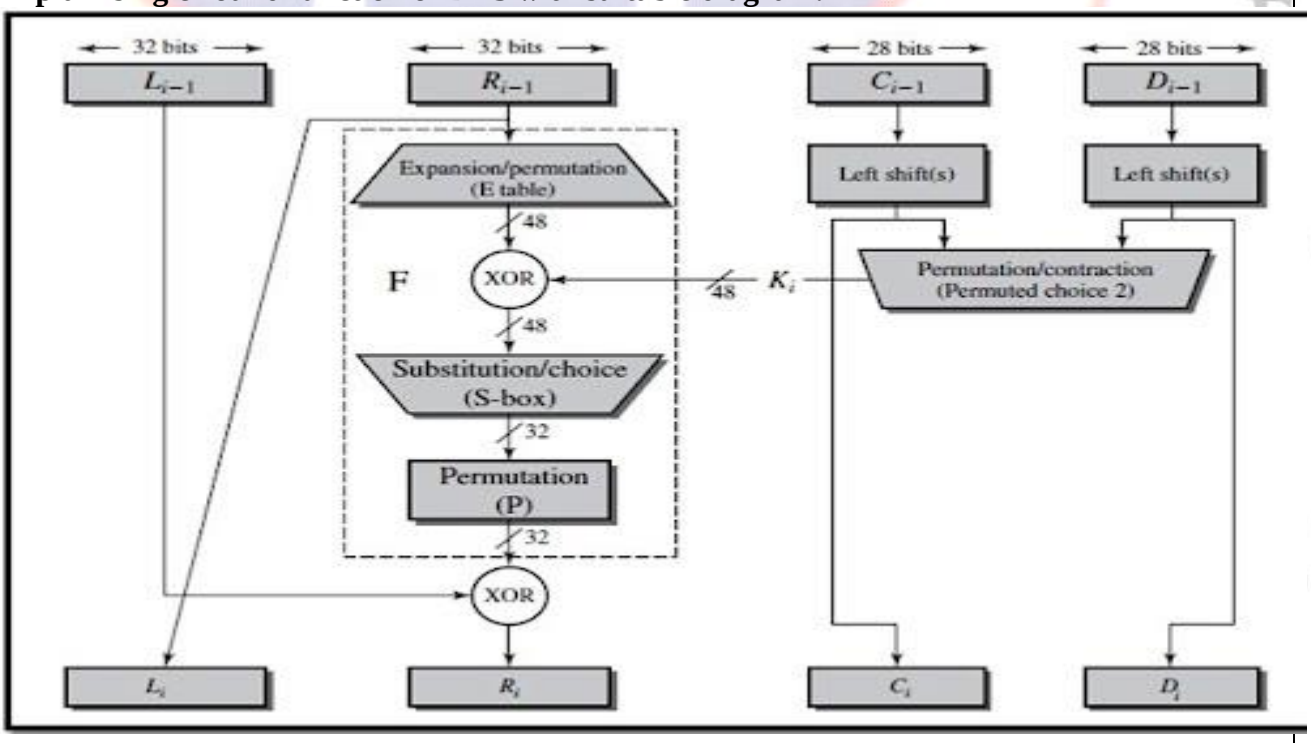
Fast Software Encryption/Decryption

- The block cipher is implemented in a software application to achieve better execution speed.

Easy Analysis

- The block cipher algorithm should be easy to analyze because it would ease in analyzing the cryptanalytic weakness and develop more strength in the algorithm.

6. **Explain single round function of DES with suitable diagram.**



- DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same

algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.

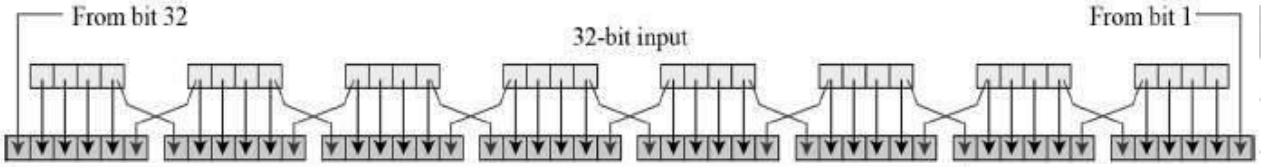
Initial and Final Permutation

- The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows –

Round Function

- The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

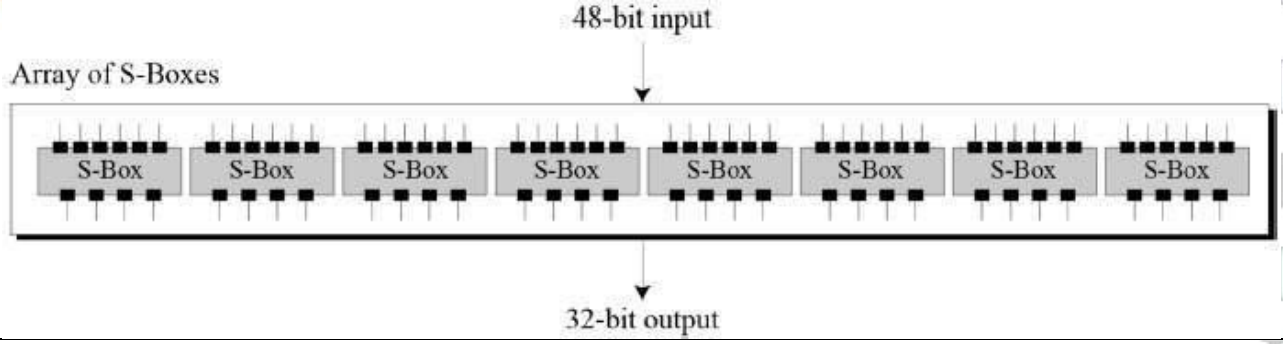
Expansion Permutation Box − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration

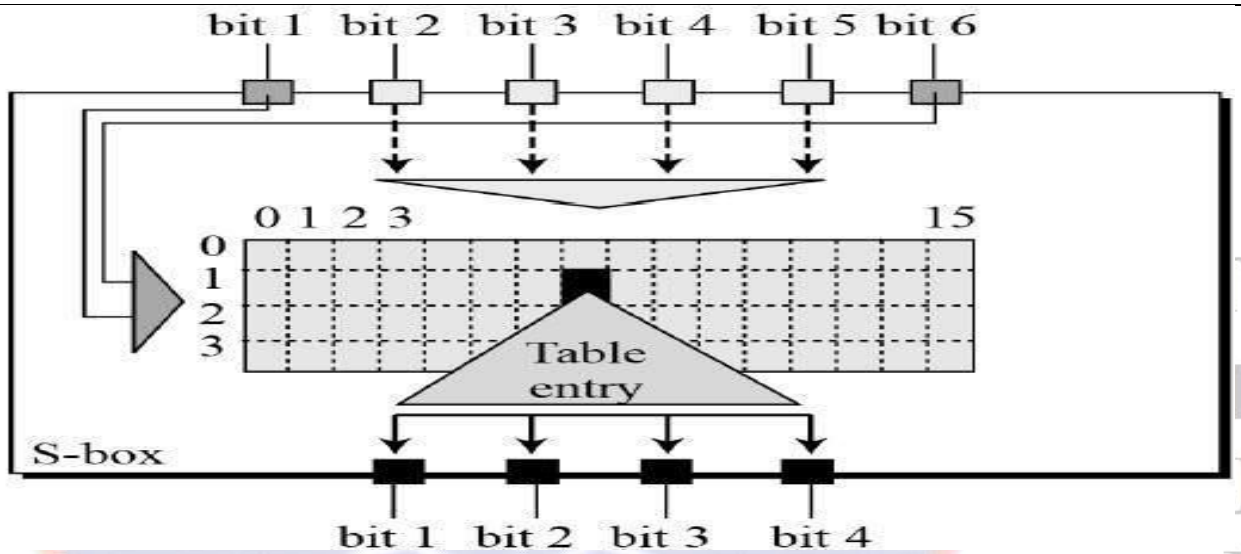


32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

XOR (Whitener). − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

Substitution Boxes. − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration –





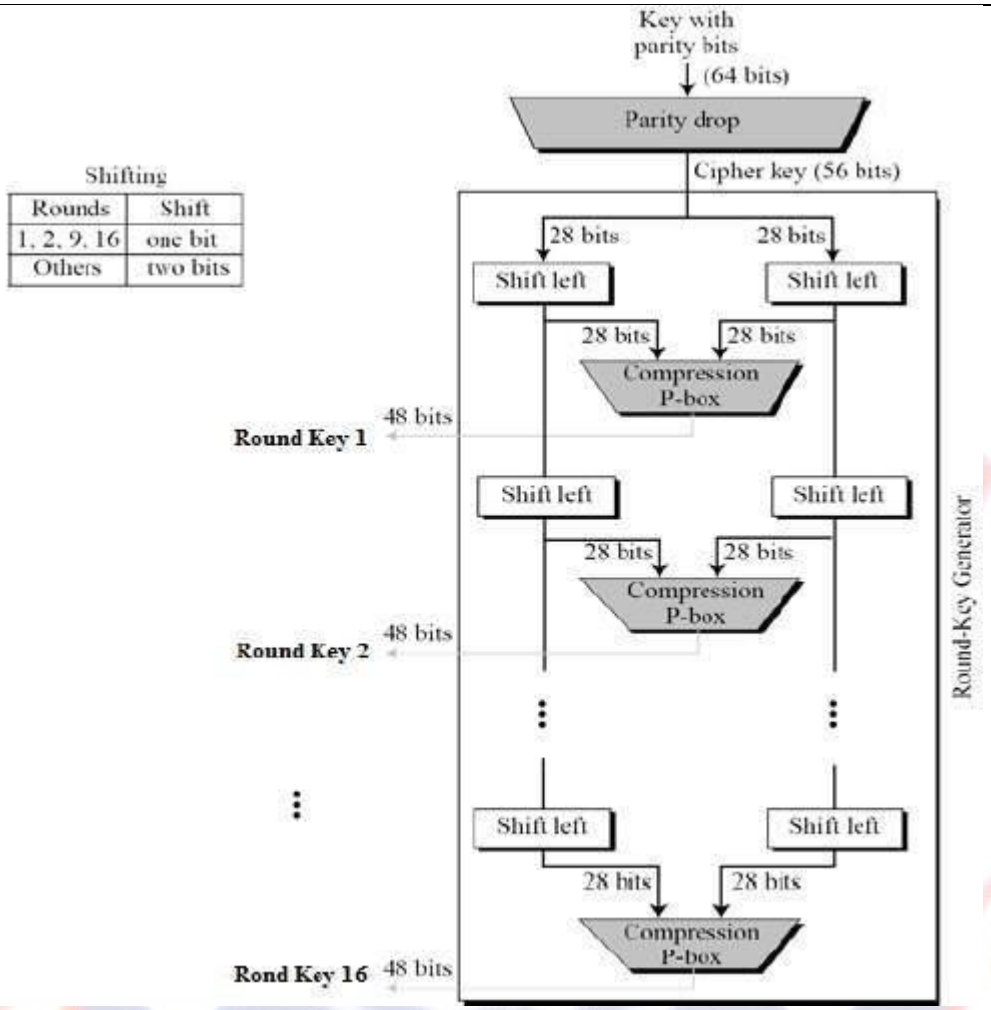
There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.

Straight Permutation *− The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:*

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

7. **Explain key generation and use of S-box in DES algorithm**

- The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –



Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits

- Parity Drop The preprocess before key expansion is a compression transposition step that we call parity bit drop. It drops the parity bits (bits 8, 16, 24, 32, …, 64) from the 64-bit key and permutes the rest of the bits according to Table 6.12. The remaining 56-bit value is the actual cipher key which is used to generate round keys. The parity drop step (a compression D-box) is shown in Table

Table: Parity-bit drop table

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

- Shift Left After the straight permutation, the key is divided into two 28-bit parts. Each part is shifted left (circular shift) one or two bits. In rounds 1, 2, 9, and 16, shifting is one bit; in the other rounds, it is two bits. The two parts are then combined to form a 56-bit part
- Compression D-box The compression D-box changes the 58 bits to 48 bits, which are used as a key for a round. The compression step is shown in Table

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

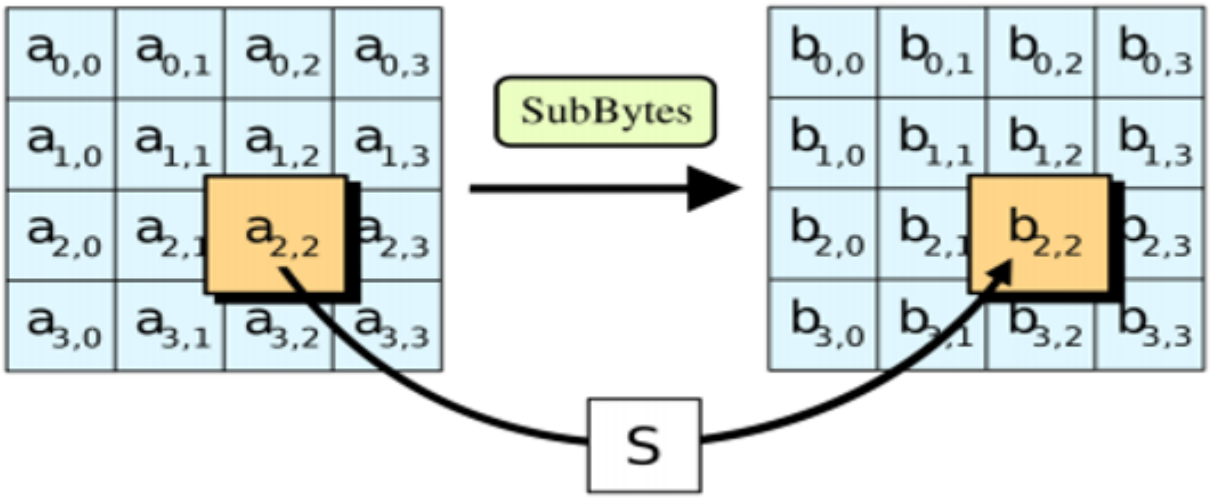
Figure - compression permutation

8. **Explain four stages of a single round in AES.**

Each round in AES algorithm consists of four steps.

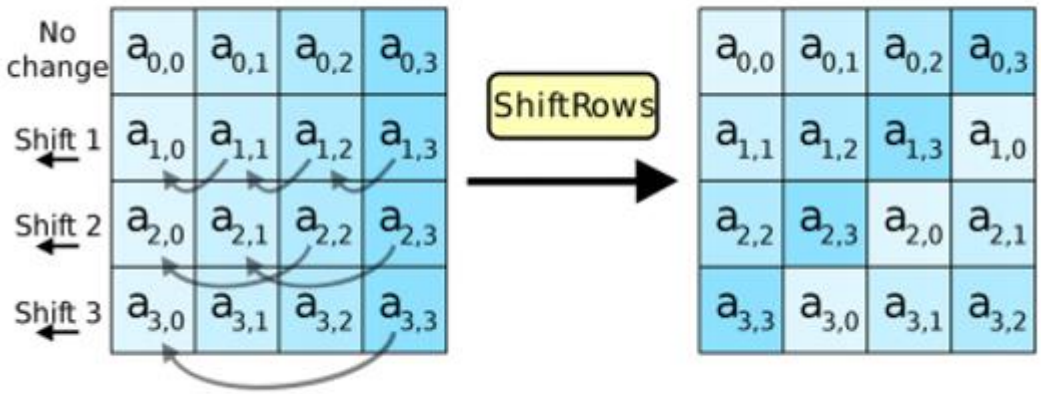
Substitution of the bytes

- In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).



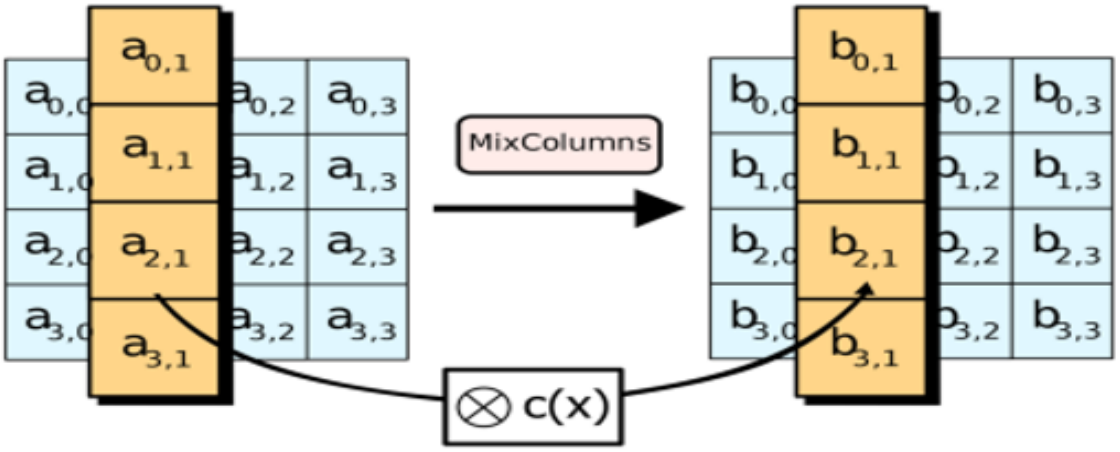
Shifting the rows

- Next comes the permutation step. In this step, all rows except the first are shifted by one, as shown below.



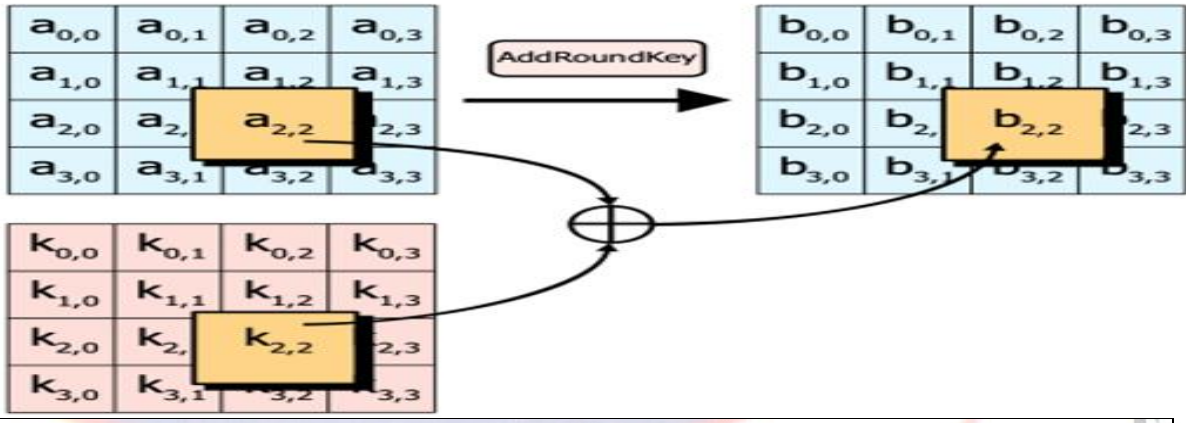
Mixing the columns

- In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.



Adding the round key

- In the final step, the message is XORed with the respective round key.



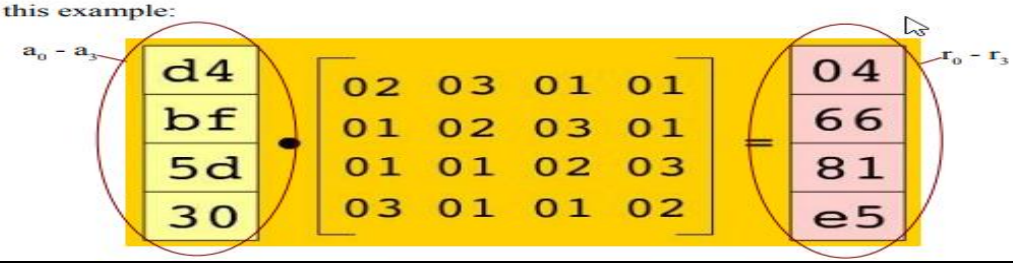
9. **Briefly describe Mix Columns and Add Round Key in AES algorithm.**

The mix columns theory is calculated using this formula[1]:

$$\begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

where r_0, r_1, r_2 and r_3 are the results after the transformation. $a_0 - a_3$ can be obtain from the matri after the data undergoes substitution process in the S-Boxes. We will now discuss the forward n column transformation. (I am assuming you know the theory for XOR gates and some other sim theories)

Let's take this example:



In this example, our $a_0 - a_3$ is equals to d4 – 30 and $r_0 - r_3$ is equals to 04 – e5. One thing to note this is that it still follows the matrix multiplication rules: row x column. Currently the matrix size looks like this:

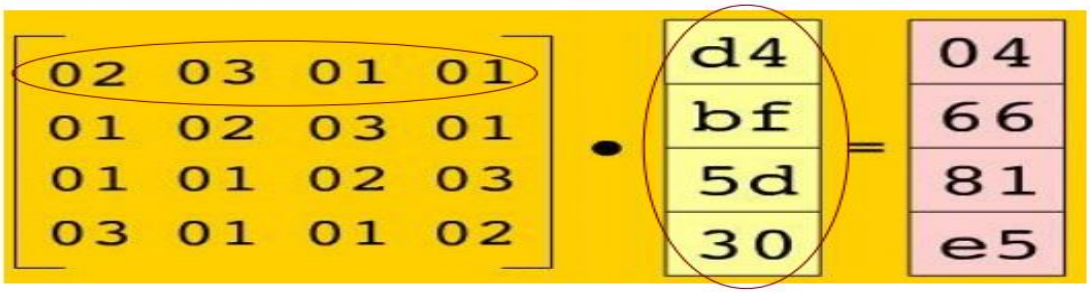
$$[4 \times 1] \cdot [4 \times 4] \neq [4 \times 1]$$

Understanding AES Mix-Columns Transformation Calculation

If you would to remember matrix idea of multiplication, to obtain [4 x 1], we need the formula to

$$[4 \times \cancel{4}] \cdot [\cancel{4} \times 1] = [4 \times 1]$$

Therefore we need to switch matrices over.



Now we are pretty much ready to calculate the answers. Like I mention early, we will multiply the rows with the column. Let's take the first row of the first matrix and multiply them with our a's values.

To get the r_0 value, the formula goes like this:

$$r_0 = \{02.d4\} + \{03.bf\} + \{01.5d\} + \{01.30\}$$

Wow. Does it not seems easy to obtain the answer? Yes, it LOOKS easy. But when it comes to calculating, apparently it isn't anymore. We will go into the steps one at a time.

1. {02.d4}

We will start with converting d4 to binary. Remember d4 is a byte so when using the Calculator program on the computer, change it to byte under Hex mode. (Qword is usable but I still prefer to change to byte just in case)

d4 = 1101 0100

Now d4 is exactly 8 bits which is good. In the case where you never get a 8 bits long characters such as 25 in Hex (converted: 100101), pad on with 0 in the front of the result until you get 8 characters of 1's and 0's. (25 ends up with 0010 0101)

Now another thing to remember, there is a rule established in the multiplication of the values as written in the book, Cryptography and Network Security[2], that multiplication of a value by x (ie. by 02) can be implemented as a 1-bit left shift followed by a conditional bitwise XOR with (0001 1011) if the leftmost bit of the original value (before the shift) is 1. We can implement this rule in our calculation.

{d4} . {02} = 1101 0100 << 1 (<< is left shift, 1 is the number of shift done, pad on with 0's)
= 1010 1000 XOR 0001 1011 (because the leftmost is a 1 before shift)
= 1011 0011 (ans)

Calculation:

```
  1010 1000
  0001 1011 (XOR)
  1011 0011
```

Now we do the same for our next set of values, {03.bf}

2. {03.bf}

Similarly, we convert bf into binary:

bf = 1011 1111

In this case, we are multiplying 03 to bf. Maybe you are starting to wonder how we are going to multiply them, or some might just multiply them directly. For my case, I followed what was suggested in the book[2], we split 03 up in its binary form.

03 = 11
= 10 XOR 01

We are now able to calculate our result.

{03} . {bf} = {10 XOR 01} . {1011 1111}
= {1011 1111 . 10} XOR {1011 1111 . 01}
= {1011 1111 . 10} XOR {1011 1111}
(Because {1011 1111} x 1[in decimal] = 1011 1111)
= 0111 1110 XOR 0001 1011 XOR 1011 1111
= 1101 1010 (ans)

{01.5d} and {01.30} is basically multiplying 5d and 30 with 1(in decimal) which we end up with the original values. There isn't a need to calculate them using the above method. But we do need to convert them to binary form.

5d = 0101 1101
30 = 0011 0000

Now, we can add them together. As they are in binary form, addition will be using XOR.

r_0 = {02.d4} + {03.bf} + {01.5d} + {01.30}
= 1011 0011 XOR 1101 1010 XOR 0101 1101 XOR 0011 0000
= 0000 0100
= 04 (in Hex)

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix} = \begin{bmatrix} 04 \\ 66 \\ 81 \\ e5 \end{bmatrix}$$

r_1 = {01.d4} + {02.bf} + {03.5d} + {01.30}

1. {02.bf}

{bf} . {02} = 1011 1111 << 1
= 0111 1110 XOR 0001 1011
= 0110 0101

2. {03.5d}

{5d} . {03} = {0101 1101 . 02} XOR { 0101 1101}
= 1011 1010 XOR 0101 1101
= 1110 0111

Therefore,

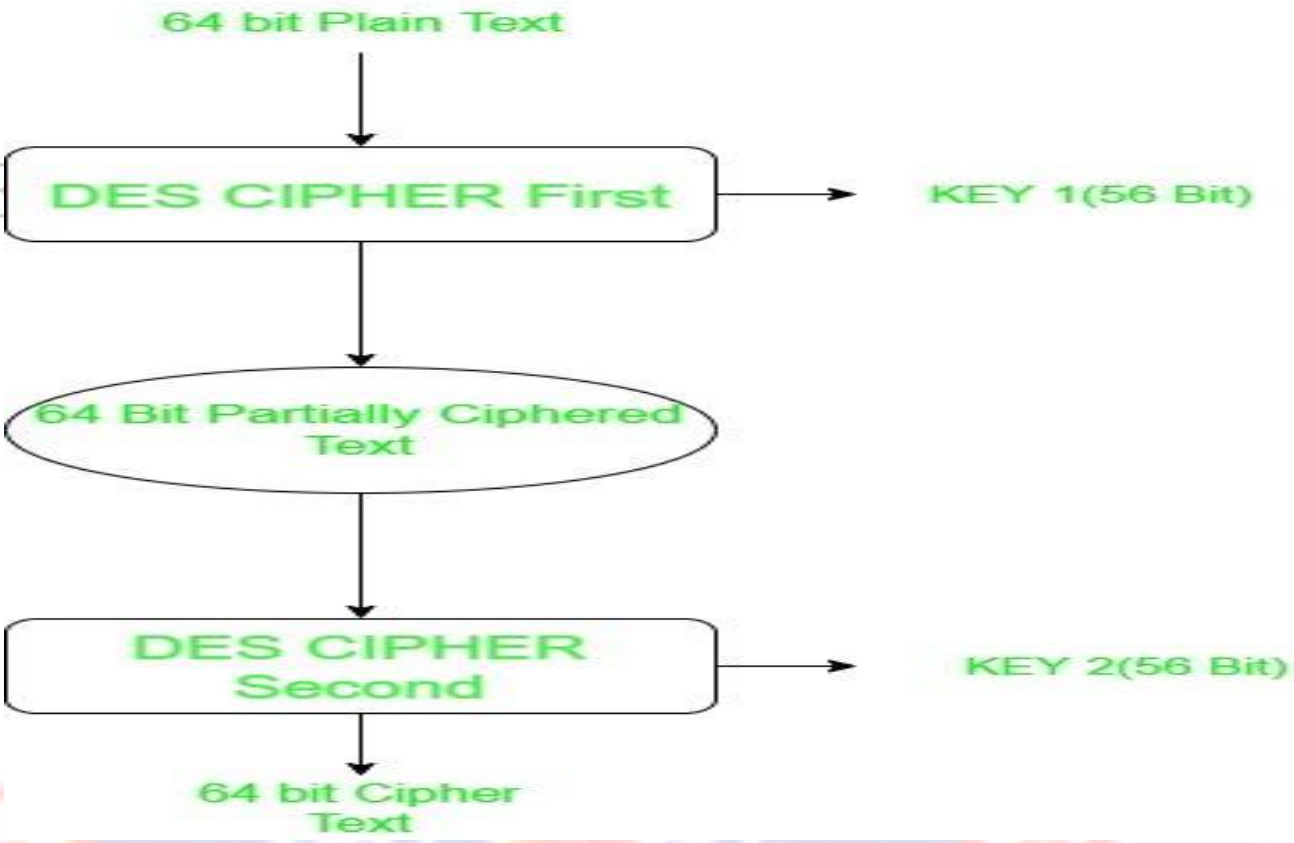
r_1 = {01.d4} + {02.bf} + {03.5d} + {01.30}
= 1101 0100 XOR 0110 0101 XOR 1110 0111 XOR 0011 0000
= 0110 0110
= 66 (in Hex)

We got our second values, 66. Do the same for the rest and you will get all the results. We no know how to calculate the mix columns. :) Happy calculating~!

10. **Explain Double DES. And Meet-in-the-Middle Attack in detail.**

- Double DES is a encryption technique which uses two instance of DES on same plain text. In both instances it uses different keys to encrypt the plain text. Both keys are required at the time of decryption. The 64 bit plain text goes into first DES instance which than converted

into a 64 bit middle text using the first key and then it goes to second DES instance which gives 64 bit cipher text by using second key.



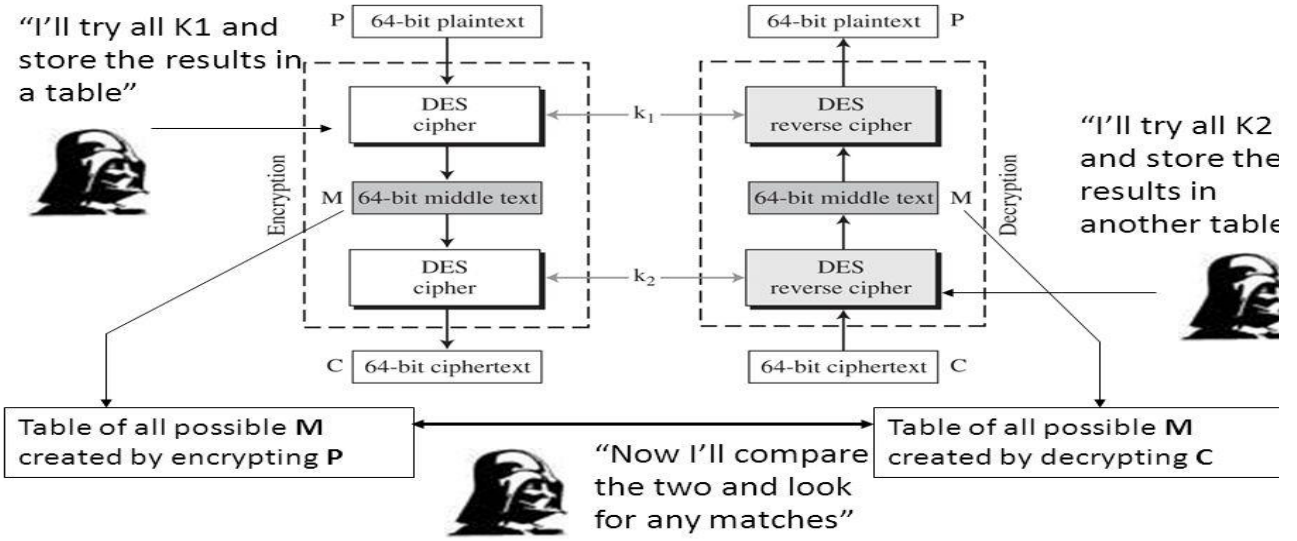
- However double DES uses 112 bit key but gives security level of 2^56 not 2^112 and this is because of meet-in-the middle attack which can be used to break through double DES.
- This attack involves encryption from one end, decryption from the other and matching the results in the middle.

Suppose cryptanalyst knows Pi and corresponding Ci.

Now, the aim is to obtain the values of K1 and K2.
For all possible values (256) of key K1, the cryptanalyst would encrypt the known plaintext by performing E(K1,P).

The cryptanalyst would store output in a table.Cryptanalyst decrypt the known ciphertext with all possible values of K2.

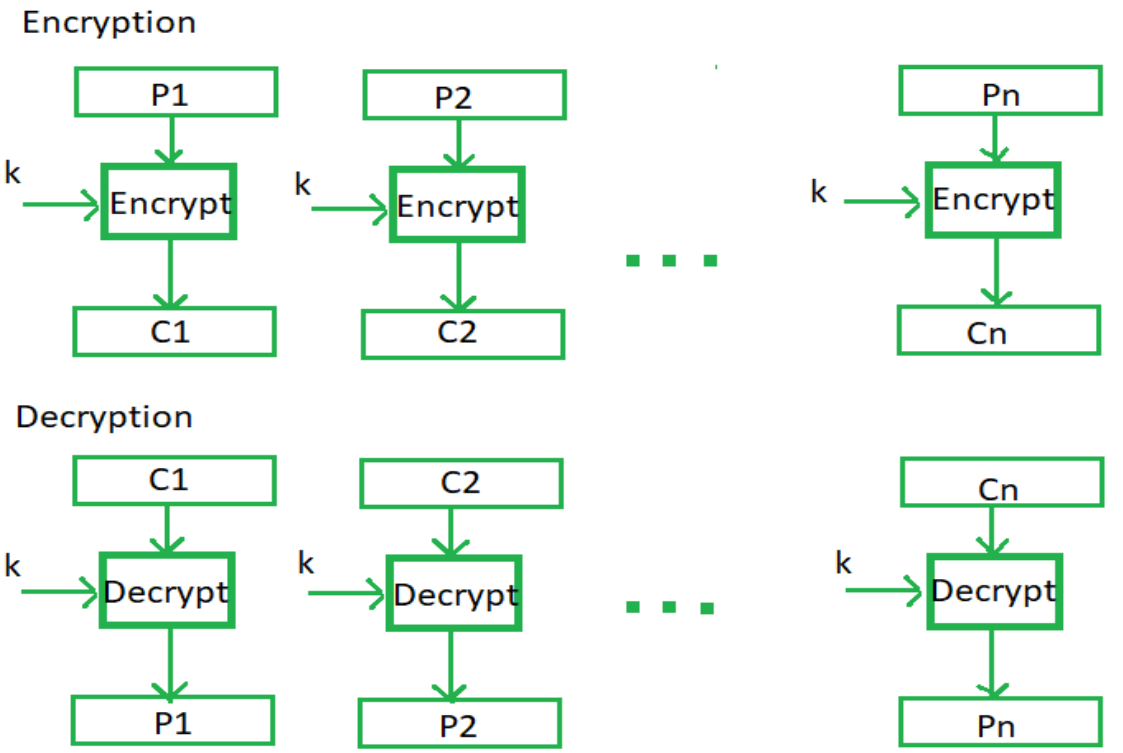
In each case cryptanalyst will compare the resulting value with the all values in the table of ciphertext.



11. **Explain Modes of Operations of block cipher.**

Electronic Code Book (ECB) –

- Electronic code book is the easiest block cipher mode of functioning. It is easier because of direct encryption of each block of input plaintext and output is in form of blocks of encrypted ciphertext. Generally, if a message is larger than b bits in size, it can be broken down into a bunch of blocks and the procedure is repeated.



Advantages of using ECB –

Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
Simple way of the block cipher.

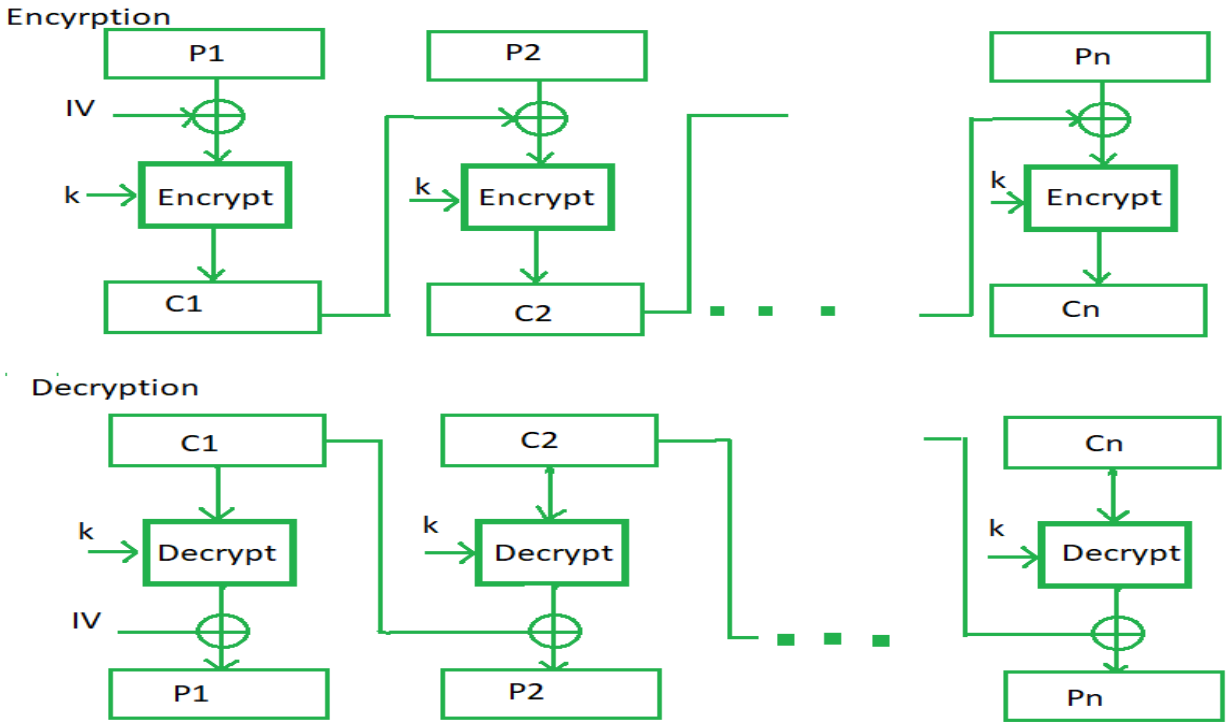
Disadvantages of using ECB –

Prone to cryptanalysis since there is a direct relationship between plaintext and ciphertext.

Cipher Block Chaining –

- Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements. In CBC, the previous cipher block is given as input to the next encryption algorithm after XOR with the original plaintext block. In a nutshell here, a cipher block is produced by encrypting an XOR output of the previous cipher block and present plaintext block.

The process is illustrated here:



Advantages of CBC –

CBC works well for input greater than b bits.

CBC is a good authentication mechanism.

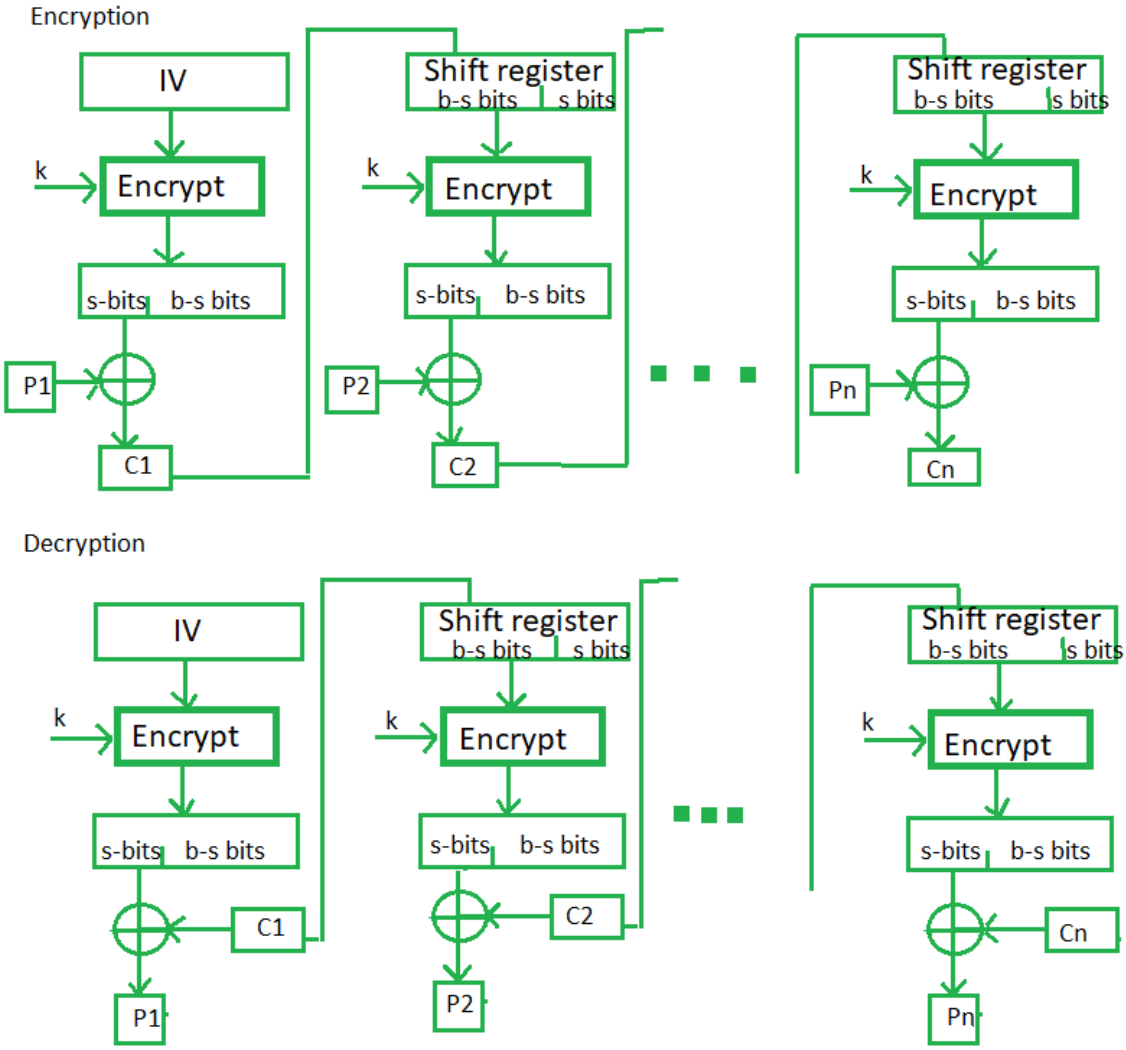
Better resistive nature towards cryptanalysis than ECB.

Disadvantages of CBC –

Parallel encryption is not possible since every encryption requires a previous cipher.

Cipher Feedback Mode (CFB) –

- In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first, an initial vector IV is used for first encryption and output bits are divided as a set of sandb-s bits the left-hand side sbits are selected and are applied an XOR operation with plaintext bits. The result is given as input to a shift register and the process continues. The encryption and decryption process for the same is shown below, both of them use encryption algorithms.

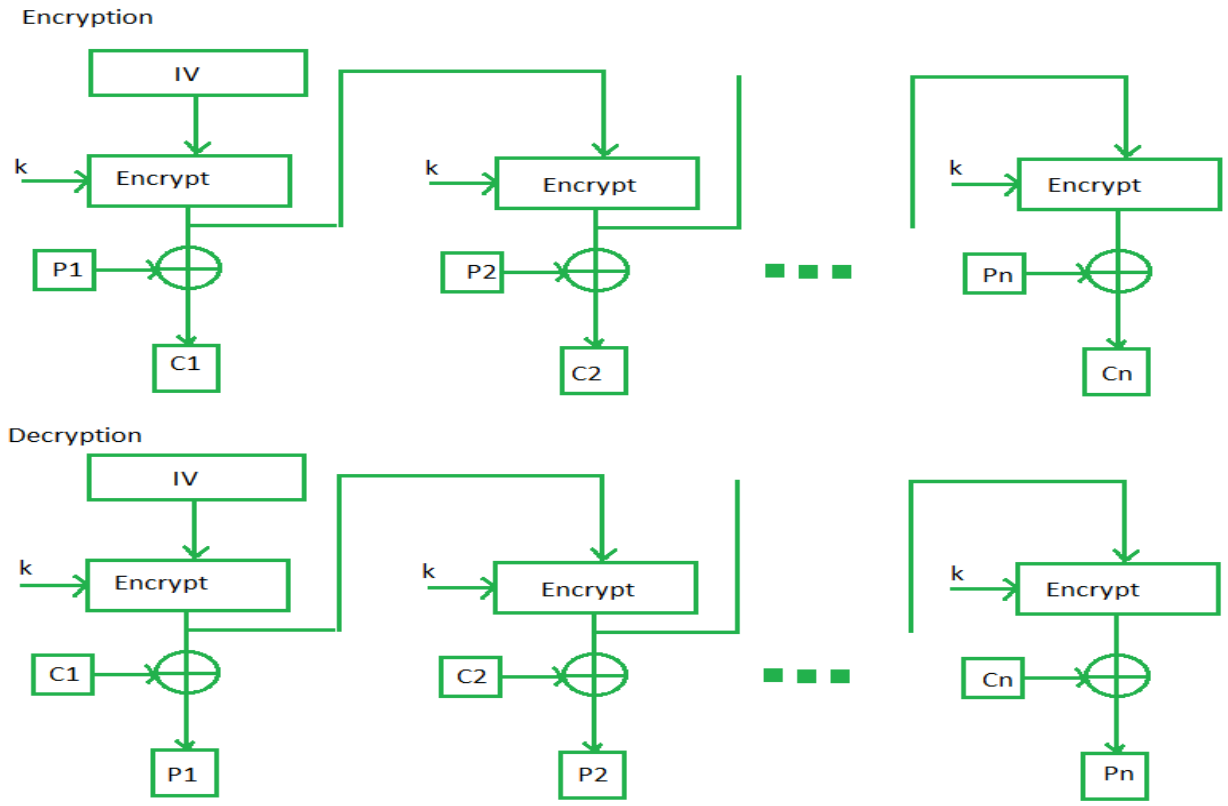


Advantages of CFB –

Since, there is some data loss due to the use of shift register, thus it is difficult for applying cryptanalysis.

Output Feedback Mode –

- The output feedback mode follows nearly the same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output. In this output feedback mode, all bits of the block are sent instead of sending selected s bits. The Output Feedback mode of block cipher holds great resistance towards bit transmission errors. It also decreases the dependency or relationship of the cipher on the plaintext.



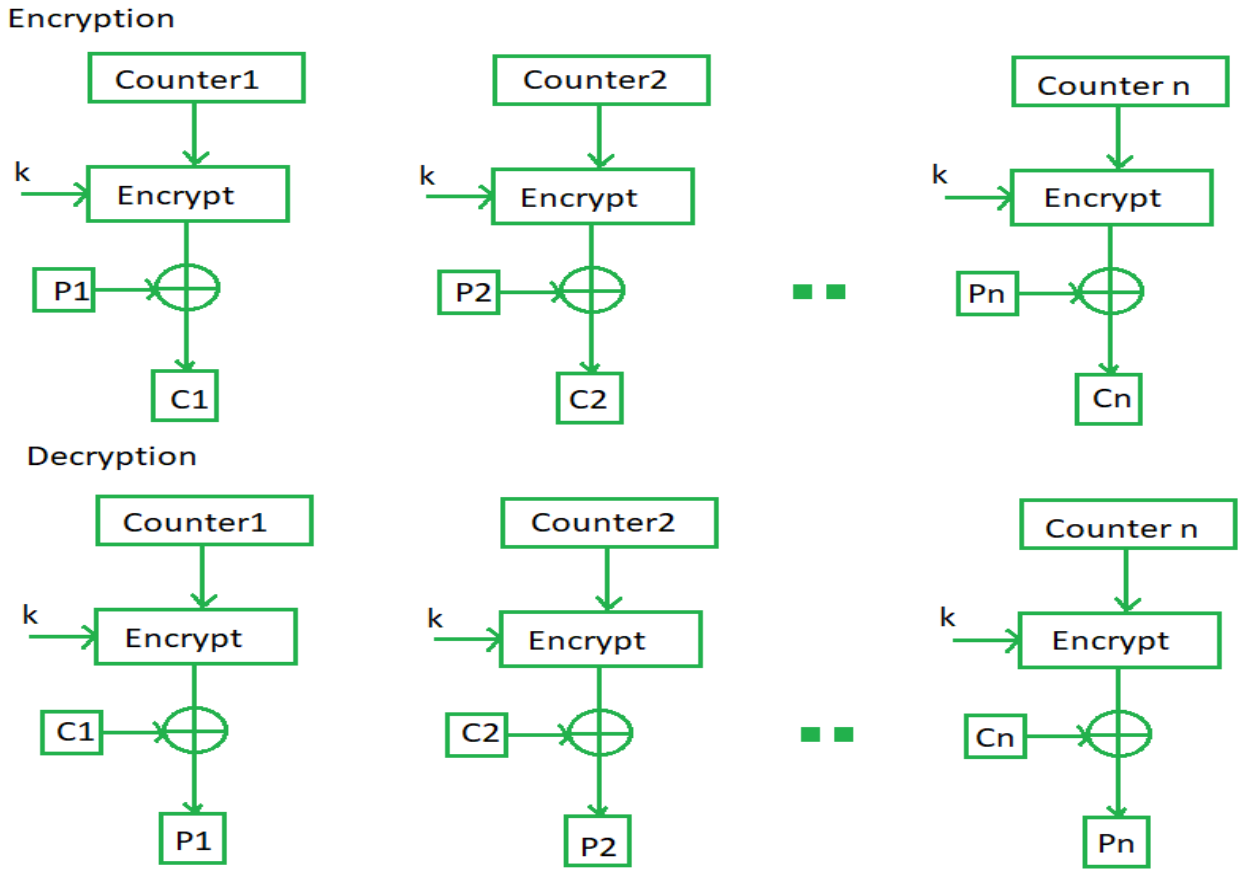
Advantages of OFB –

In the case of CFB, a single bit error in a block is propagated to all subsequent blocks. This problem is solved by OFB as it is free from bit errors in the plaintext block.

Counter Mode –

- The Counter Mode or CTR is a simple counter-based block cipher implementation. Every time a counter-initiated value is encrypted and given as input to XOR with plaintext which results in ciphertext block. The CTR mode is independent of feedback use and thus can be implemented in parallel.

Its simple implementation is shown below:



Advantages of Counter –
Since there is a different counter value for each block, the direct plaintext and ciphertext relationship is avoided. This means that the same plain text can map to different ciphertext. Parallel execution of encryption is possible as outputs from previous stages are not chained as in the case of CBC.

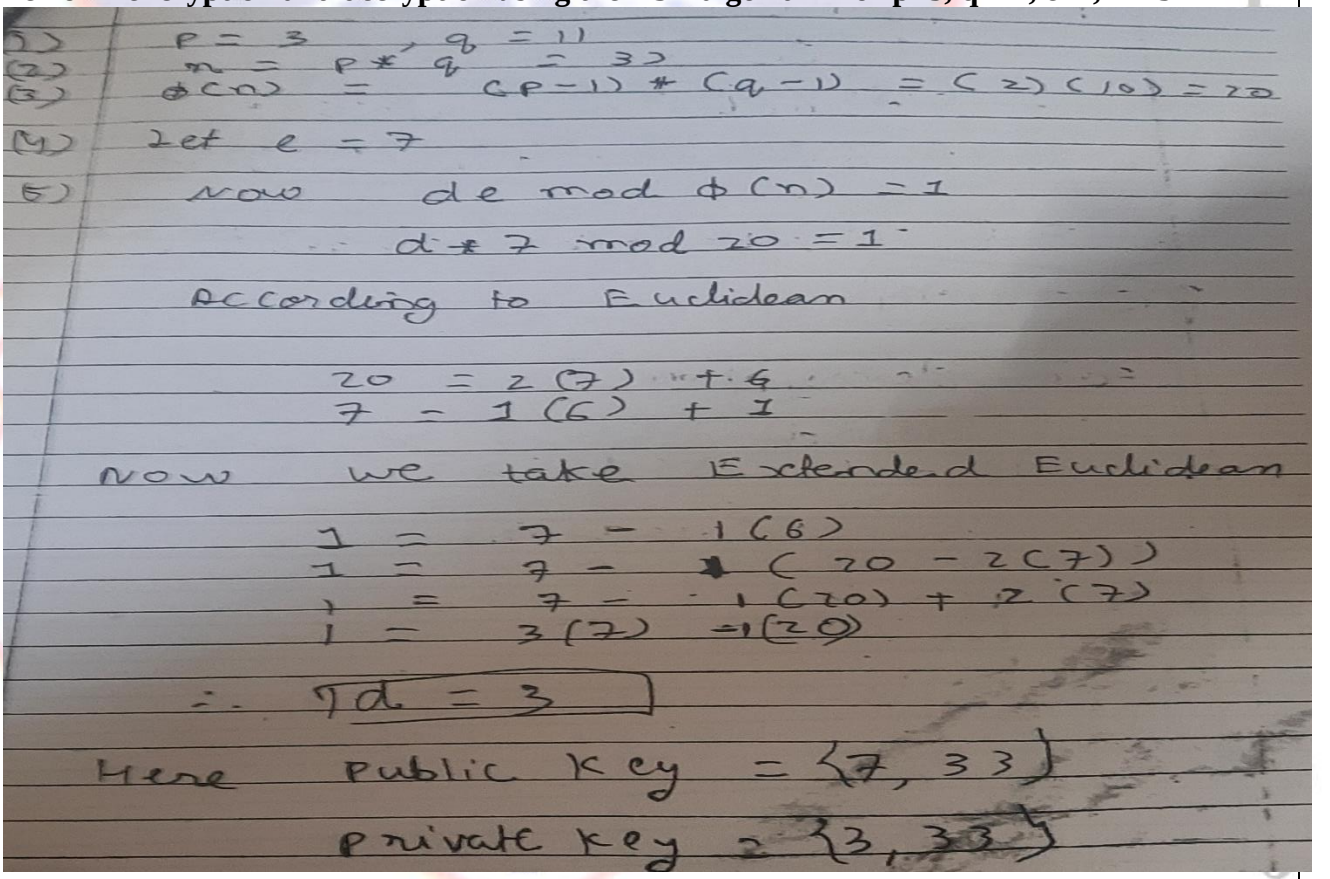
12. **Distinguish between Symmetric encryption and Asymmetric encryption using suitable example**

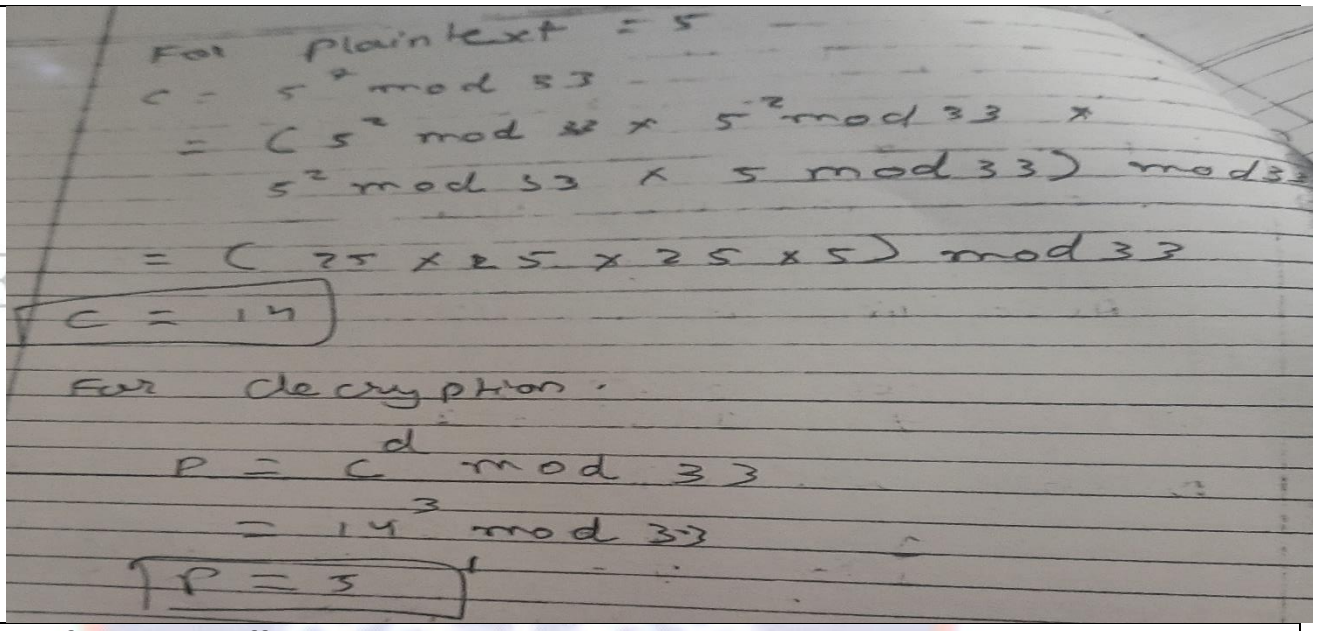
Symmetric Key Encryption:

- Encryption is a process to change the form of any message in order to protect it from reading by anyone. In Symmetric-key encryption the message is encrypted by using a key and the same key is used to decrypt the message which makes it easy to use but less secure. It also requires a safe method to transfer the key from one party to another.
- Asymmetric Key Encryption:
- Asymmetric Key Encryption is based on public and private key encryption technique. It uses two different key to encrypt and decrypt the message. It is more secure than symmetric key encryption technique but is much slower.

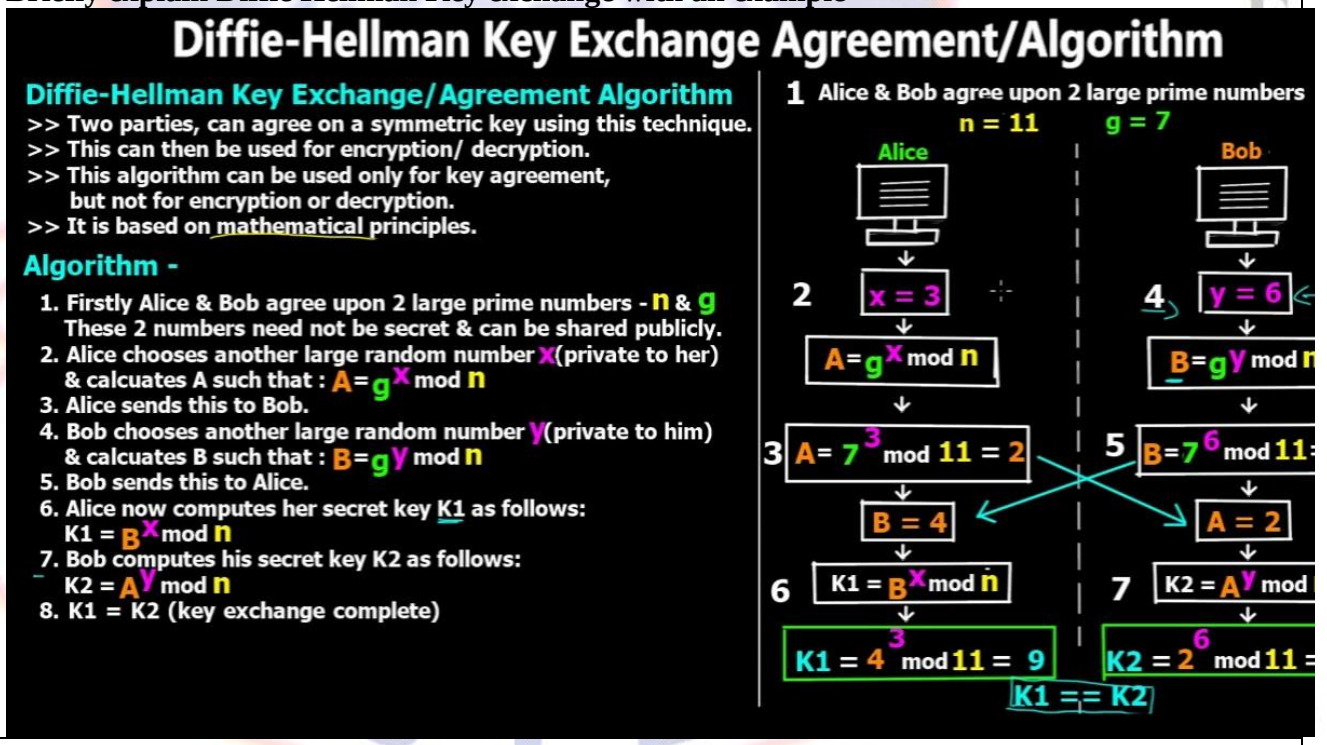
Characteristic	Symmetric Cryptography	Asymmetric Cryptograph
Key used for encryption/decryption	Same key is used	One key is used for encryption and another for decryption
Speed of encryption/decryption	Very fast	Slower
Size of resulting encrypted text	Usually same as or less than the original plaintext size	More than the original plaintext size
Known keys	Both parties should know the key in symmetric key encryption	One of the keys is known by the two parties in public key encryption
Usage	Confidentiality	Confidentiality, Digital signat

13. **Give the steps of RSA algorithm and solve below example:**
Perform encryption and decryption using the RSA algorithm for p=3, q=11, e=7, M=5





14. **Briefly explain Diffie Hellman Key exchange with an example**



15. **Discuss Man in Middle Attack**

- The Diffie-Hellman key exchange protocol is exposed to man-in-the-middle (MITM) attack. An attacker can interpose Alice's public value and send his own calculated public value to Bob. When Bob replies back his public value, attacker again replaces it with his own public value and sends it to Alice

Alice	Carol (Intruder)	Bob
p=17 and α= 4 are known to all		
Choose $k_{pri,A}$ =a = 7		Choose $k_{pri,B}$ =b = 8
Alice's public key: $k_{pub,A}$ = A= α^a mod p =		Bob's public key: $k_{pub,B}$ = B=α^b mod p =
Send A to Bob; intercepted by Carol		
	Send B to Alice; intercepted by Carol	
	Carol chooses c=6; computes A' = B' =α^c mod p	
	Carol sends A' to Bob as if it is A from Alice	
Carol sends B' to Alice as if it is from Bob		
Alice derives the shared secret key as K1= B'^a mod p	Carol derives K1= A^c mod p, K2= B^c mod p,	Bob derives the shared secret key as K2= A'^b mod p
Session 1 established with key K1: verify that Alice and Carol have derived the same key K1		
	Session 2 established with key K2; verify that Carol and Bob have derived the same key K2	

16. **What are the characteristics of hash function? Also explain basic techniques of hash algorithm**

- Hash functions are extremely useful and appear in almost all information security applications.
- A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.
- Values returned by a hash function are called message digest or simply hash values

Properties of Hash Functions

In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

1) It should be computationally hard to reverse a hash function.
In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
This property protects against an attacker who only has a hash value and is trying to find the input.

2) Given an input and its hash, it should be hard to find a different input with the same hash.
In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

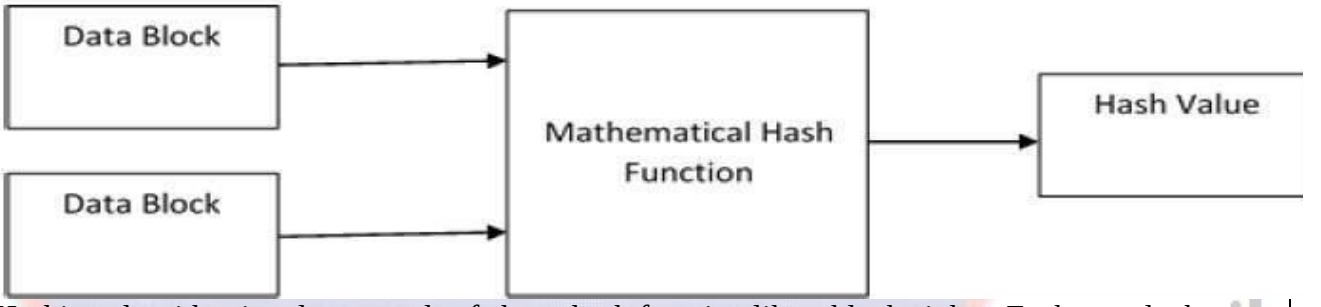
3) Collision Resistance
This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).

Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.

This property makes it very difficult for an attacker to find two input values with the same hash.

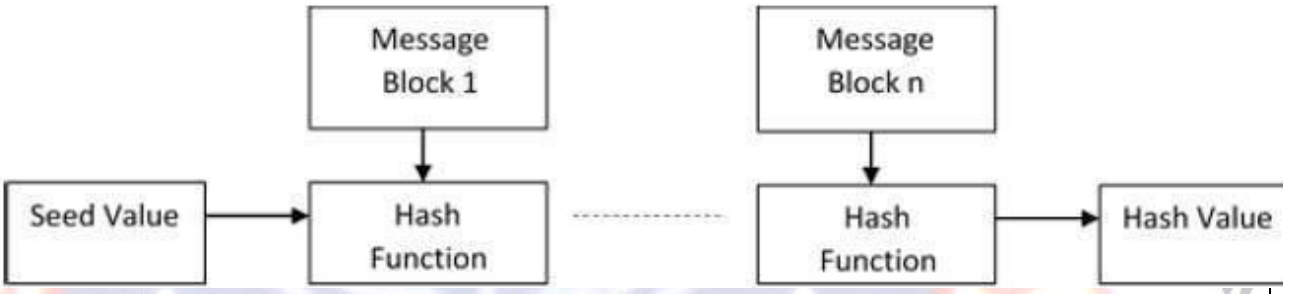
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function –



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration –

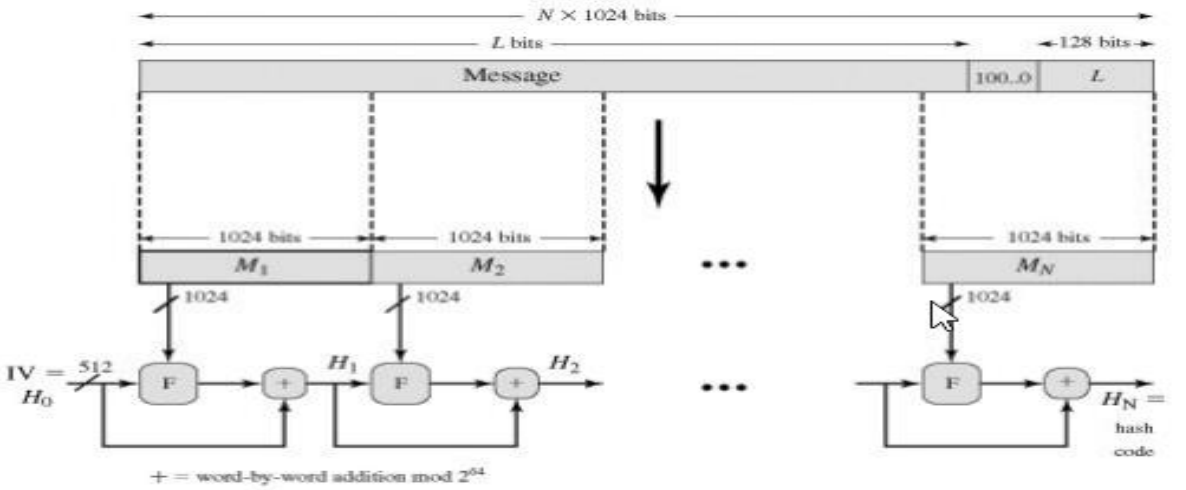


Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an avalanche effect of hashing.

17. **Explain SHA512 Algorithm**

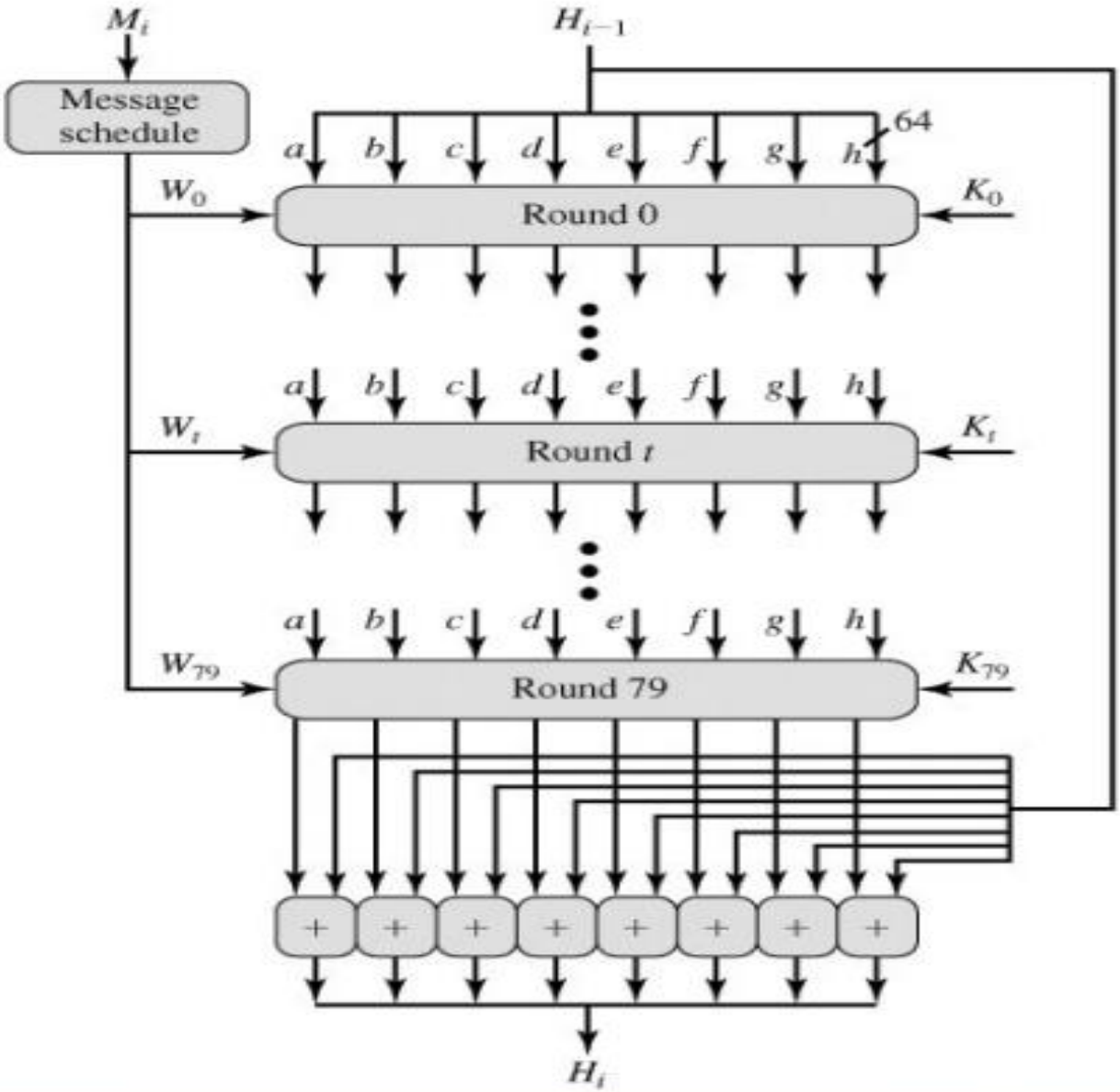
- The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest.
- The input is processed in 1024-bit blocks. Figure 12.1 depicts the overall processing of a message to produce a digest.

Figure 12.1
Message Digest Generation Using SHA-512



Step 1: Append padding bits.

	The message is padded so that its length is congruent to 896 modulo 1024 [length 896 (mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1-bit followed by the necessary number of 0-bits **Step 2: Append length.** A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 12.1, the expanded message is represented as the sequence of 1024-bit blocks M1, M2,..., MN, so that the total length of the expanded message is N x 1024 bits. **Step 3: Initialize hash buffer.** A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values): a = 6A09E667F3BCC908 b = BB67AE8584CAA73B c = 3C6EF372FE94F82B d = A54FF53A5F1D36F1 e = 510E527FADE682D1 f = 9B05688C2B3E6C1F g = 1F83D9ABFB41BD6B h = 5BE0CDI9137E2179 These values are stored in big-endianformat, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers. **Step 4:** Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 12.1. The logic is illustrated in Figure 12.2.



Each round also makes use of an additive constant Kt where 0 t 79 indicates one of the 80 rounds.
These words represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data.
The output of the eightieth round is added to the input to the first round (Hi-1)to produce Hi. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in Hi-1 using addition modulo 264.

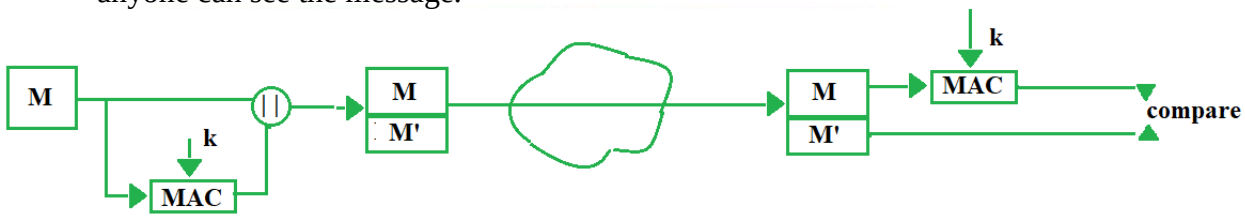
Step 5: Output.

After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.
We can summarize the behavior of SHA-512 as follows:
H0 = IV
Hi = SUM64(Hi-1, abcdefghi)
MD = HN
where
IV = initial value of the abcdefgh buffer, defined in step 3 abcdefghi = the output of the last round of processing of the ith message block N = the number of blocks in the message (including padding and length fields) SUM64 = Addition modulo 264 performed separately on each word of the pair of inputs MD = final message digest value

18. **Write a short note on Message Authentication Code**

- MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

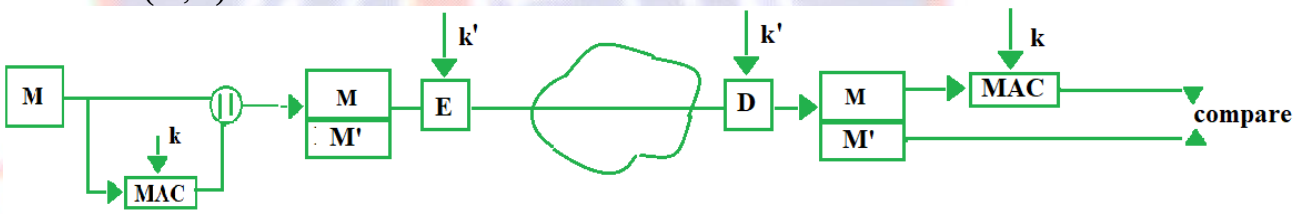
- Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.
- MAC without encryption – This model can provide authentication but not confidentiality as anyone can see the message.



Internal Error Code

In this model of MAC, sender encrypts the content before sending it through network for confidentiality. Thus this model provides confidentiality as well as authentication.

M' = MAC(M, k)

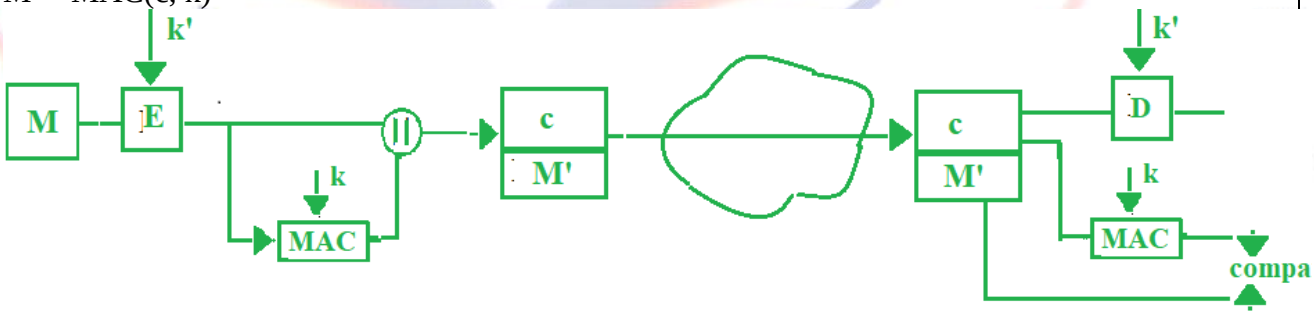


External Error Code

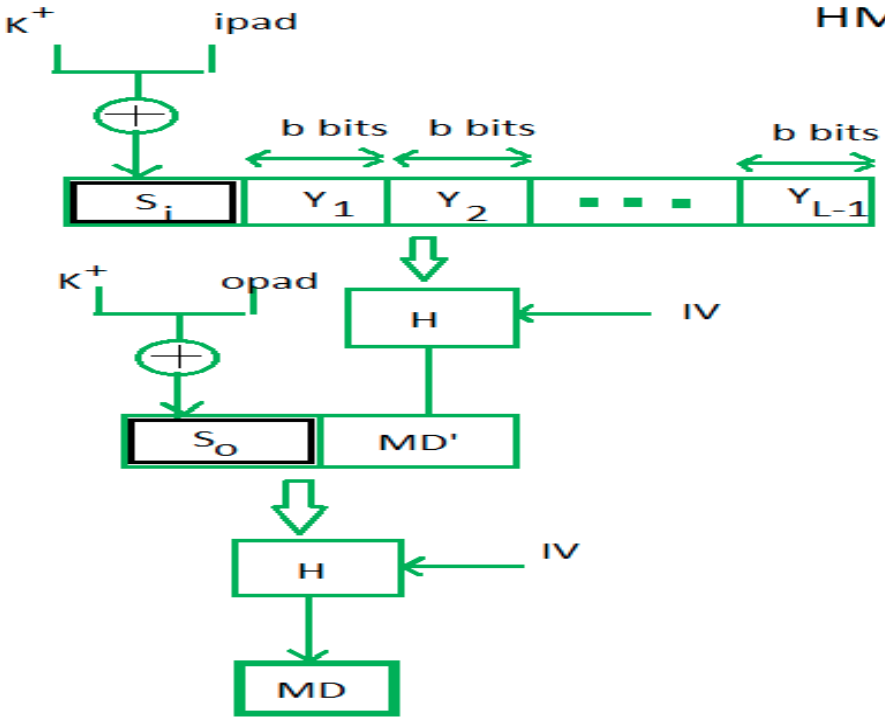
For cases when there is an alteration in message, we decrypt it for waste, to overcome that problem, we opt for external error code. Here we first apply MAC on the encrypted message 'c' and compare it with received MAC value on the receiver's side and then decrypt 'c' if they both are same, else we simply discard the content received. Thus it saves time.

c = E(M, k')

M' = MAC(c, k)



19. **Explain HMAC algorithm**



The formula for working with HMAC goes as follows

HMAC=hasfunc (secretkey message)

Firstly, the authentication function is of three types, namely

·Message encryption

·Message Authentication code

·Hash Functions.

In HMAC the function of hash is applied with a key to the plain text. But before application of the function, there is a need to calculate S bits and then affix it to plain text and after that apply hash function. For a generation of those S bits, there is the use of a key that is shared between the sender and receiver.

In Brief, they can be said as,

Select K.

If K < b, pad 0's on left until k is equal to b.And K is between 0 and b (0 < K < b)

EXOR K with ipad equivalent to b bits producing S1 bits.

Append S1 with plain text M

Apply SHA-512 on (S1 || M)

Pad n-bits until length is equal to b-bits
EXOR K with opad equivalent to b bits producing S2 bits.
Append S2 with output of step 5.
Apply SHA-512 on step 7 to output n-bit hash code.
The Procedure in detail can be explained as,

K is generated using K by padding O's on the left side until the dimensions of K becomes b bits. B bits are the size of the blocks of the plain text. Ipad and opad are the two padding bits that are defined well in advance. And most importantly all these are finished before the hash function is applied.

Ipad – 00110110 opad – 01011100

Now there is a need for the calculation of S bits

S1 bits are obtained when K along with ipad is EXORed . The S1 resulting here is equal to b bits. This is obvious since both K and ipad are made up of b bits. Let the plain text message be taken as P with which we have to append S1.

From S1 to Pm each block is in b bits.

And M here is the number of blocks of plain text.

and p0 is a block of plain text with b, being the block size of plain text

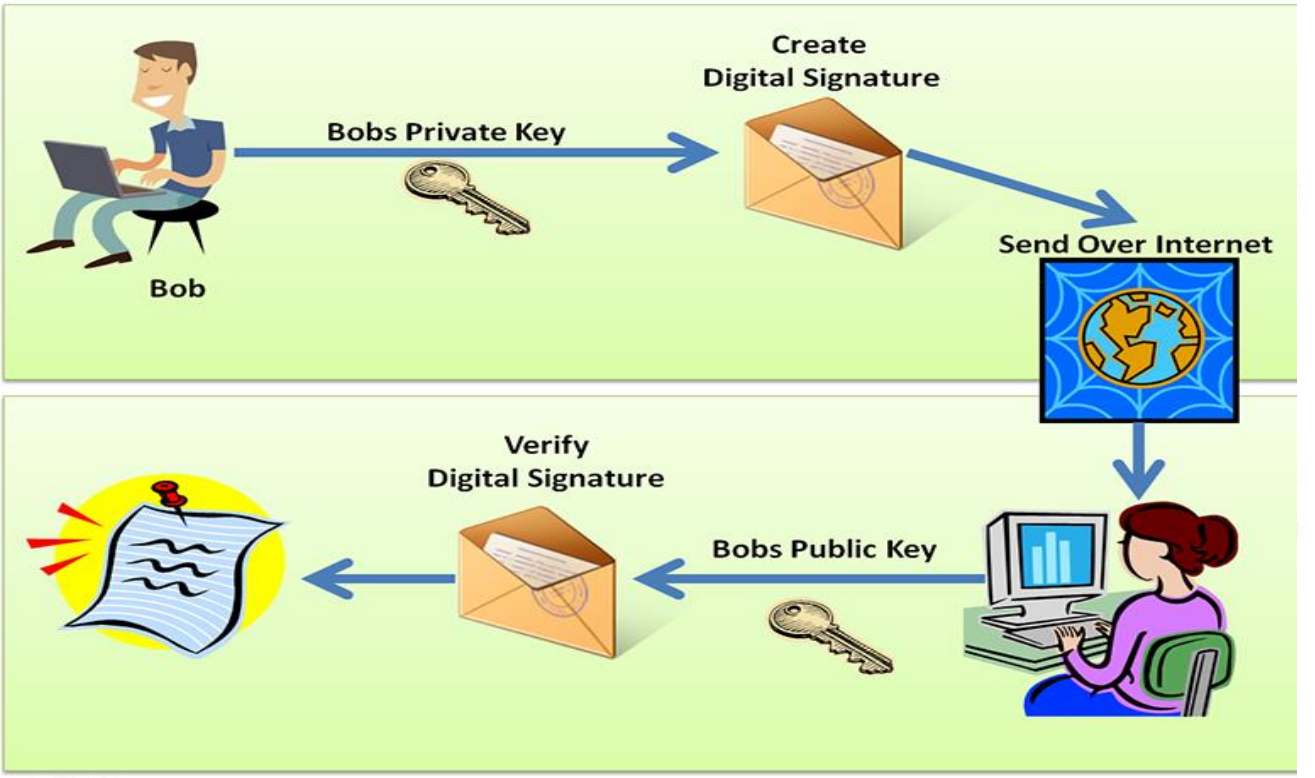
The HASH algorithm is to be applied hereafter attaching S1 to Plain text.

The resultant thus is a hash code of n-bit.

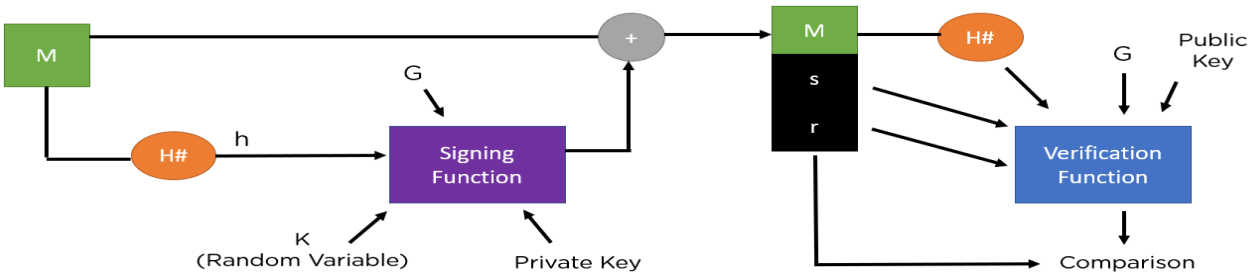
5) **ADVANTAGES**

HMACs are considered to be secured one by many; besides this, they have many advantages like,

HMACs are considered to be good high-performance systems. It is because of the hash functions used, which are quick both in terms of calculation as well as verification.
Since it uses hashing twice, it is great for cryptanalysis attacks.

	6) **DISADVANTAGES** Though there are many advantages, HMACs are not shielded from any disadvantages. Some of them are, As HMAC makes use of the shared key, If there is a compromise in the key of sender or receiver, it makes the job of creating unauthorised messages easier for attackers. There is also a need for periodic refreshments of keys, which adds to its disadvantage.
20.	**What is digital signature? Explain its use with the help of example** • A digital signature uses asymmetric encryption, also called public-key cryptography, a method that uses a key pair system. There is a set of the key where one is called the private key accessible only to the sender to encrypt the signature and the public key to decrypt the signature, available to all the people who receive data. The digital signature process involves the following steps. • The sender of the data, also called the signer generates two keys – a public key which he shares with the receiver, and a private key which he keeps with himself. One of the popular and secure digital signature algorithm to generate these keys is the RSA Algorithm. • Digital signature cryptography: The signer generates a unique value called a hash using a mathematical/hashing algorithm from the digital data file. Hash is an alpha-numeric string of fixed length. It is then encrypted using the key only known to the signer i.e. the private key. This encrypted hash is the digital signature that is appended to the document and is electronically sent to the receiver with the public key. • In this case, only the signature is encrypted, not the data in the digital document. It is more efficient and economical to encrypt a small digest rather than the whole message/data. • Digital signature verification: The receiver of the digital file decrypts the digital signature using the public key that proves the authenticity of the data file. He then generates his hash using the same hashing algorithm. If the two hash match, it certifies the integrity of the data which means that the data has not tampered in transit. 

21. **Briefly explain Digital Signature algorithm**



Steps in DSA Algorithm

Keeping the image above in mind, go ahead and see how the entire process works, starting from creating the key pair to verifying the signature at the end.

1. Key Generation

- You first choose a prime number q, which is known as the prime divisor.
- Another prime number, p, is chosen such that p-1 mod q = 0.
- Choose an integer g (1<g<p), satisfying the two conditions, g**q mod p = 1 and g = h**((p–1)/q) mod p
- x is our private key, and it is a random integer such that 0 < x < q.
- y is our public key, and you can calculate it as y = gx mod p.
- Now the private key package is {p,q,g,x}.
- The public key package is {p,q,g,y}.

2. Signature Generation

- It passes the original message (M) through the hash function (H#) to get our hash digest(h).
- It passes the digest as input to a signing function, whose purpose is to give two variables as output, s, and r.
- Apart from the digest, you also use a random integer k such that 0 < k < q.
- To calculate the value of r, you use the formula r = (gk mod p) mod q.
- To calculate the value of s, you use the formula s = [K-1(h+x . R)mod q].
- It then packages the signature as {r,s}.
- The entire bundle of the message and signature {M,r,s} are sent to the receiver.

3. Signature Verification

- You use the same hash function (H#) to generate the digest h.
- You then pass this digest off to the verification function, which needs other variables as parameters too.
- Compute the value of w such that: s*w mod q = 1
- Calculate the value of u1 from the formula, u1 = h*w mod q
- Calculate the value of u2 from the formula, u2 = r*w mod q
- The final verification component v is calculated as v = [((gu1 . yu2) mod p) mod q].
- It compares the value of v to the value of r received in the bundle.
- If it matches, the signature verification is complete.

22. **(A) Explain the key generation in AES algorithm**
(B) Explain Byte substitution and Shift row operation of AES in detail

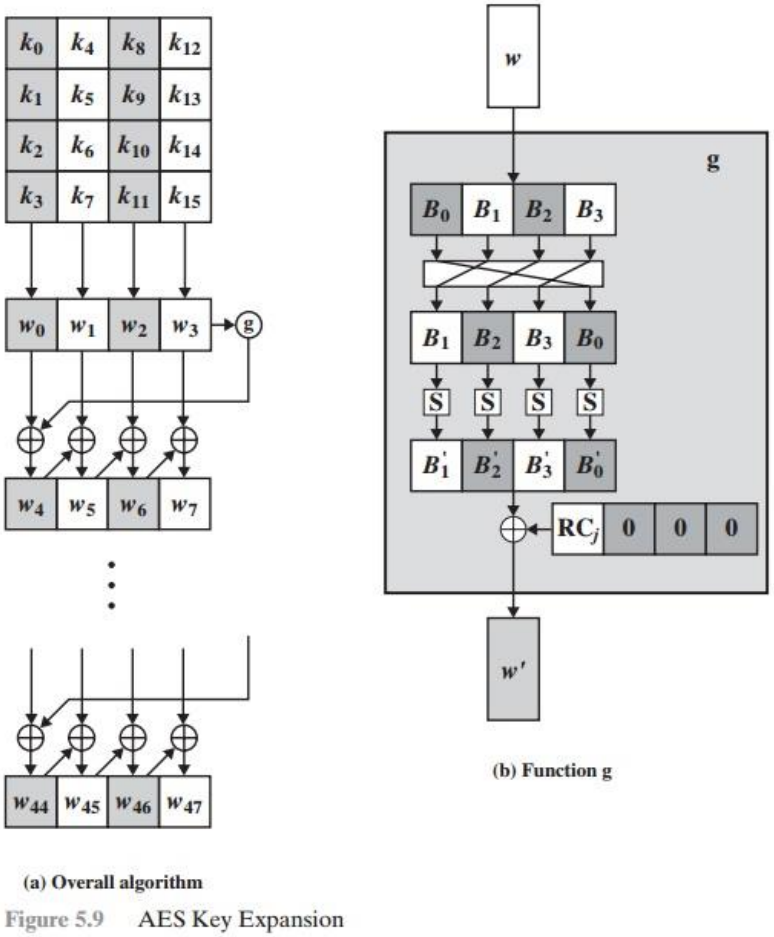


Figure 5.9 AES Key Expansion

The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The pseudocode on the next page describes the expansion.

The key is copied into the first four words of the expanded key. The remain- der of the expanded key is filled in four words at a time. Each added word w[i] depends on the immediately preceding word, w[i - 1], and the word four posi- tions back, w[i - 4]. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. Figure 5.9 illustrates the generation of the expanded key, using the symbol g to represent that complex function. The function g consists of the following subfunctions.

1. RotWord performs a one-byte circular left shift on a word. This means that an input word [B0, B1, B2, B3] is transformed into [B1, B2, B3, B0].

2. SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 5.2a).

3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

The round constant is a word in which the three rightmost bytes are always 0. Thus, the effect of an XOR of a word with Rcon is to only perform an XOR on the left- most byte of the word. The round constant is different for each round and is defined as Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1, RC[j] = 2 RC[j -1] and with multiplica- tion defined over the field GF(28). The values of RC[j] in hexadecimal are

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

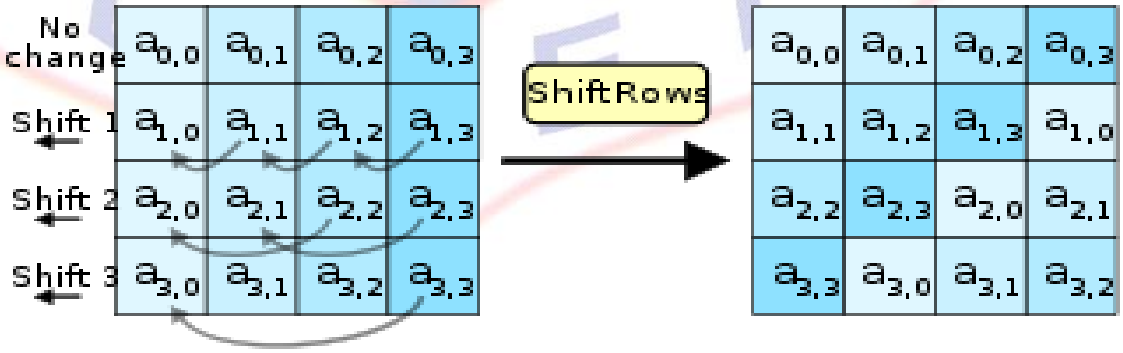
For example, suppose that the round key for round 8 is

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:

i (decimal)	temp	After RotWord	After SubWord	Rcon (9)	After XOR with Rcon	w[i−4]	w[i] = temp ⊕ w[i−4]
36	7F8D292F	8D292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Shift Rows in AES:



- This operation is a simple substitution that converts every byte into a different value. AES defines a table of 256 values for the substitution. You work through the 16 bytes of the state array, use each byte as an index into the 256-byte substitution table, and replace the byte with the value from the substitution table. Because all possible 256 byte values are present in the table, you end up with a totally new result in the state array, which can be restored to its original contents using an inverse substitution table. The contents of the substitution table are not arbitrary; the entries are computed using a mathematical formula but most implementations will simply have the substitution table stored in memory as part of the design.

AES (Advanced Encryption Standard)

1. SubBytes / Substitute Bytes

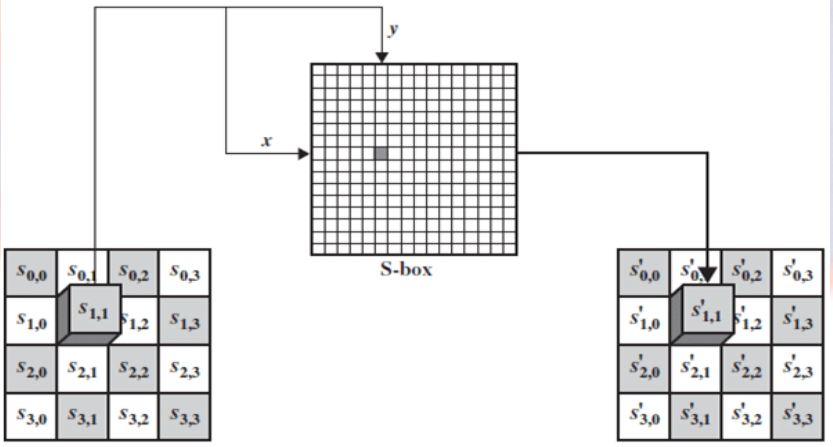
- AES defines a 16 x 16 matrix of byte values, called an S-box, that contains a permutation of all possible 256 8-bit values.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

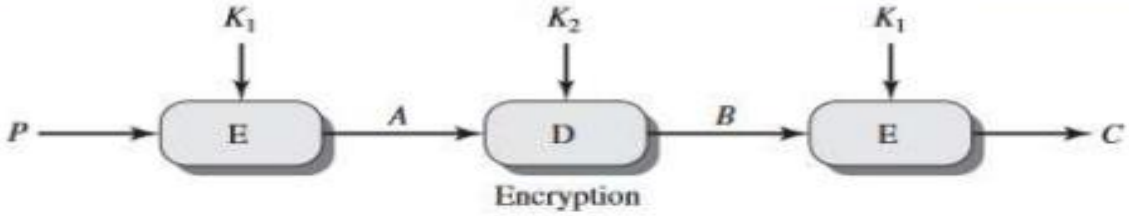


23. **Briefly explain Triple DES with two keys**

- It uses three stages of DES for encryption and decryption. The 1st, 3rd stage use K1K1 key and 2nd stage use K2K2 key.
- To make triple DES compatible with single DES, the middle stage uses decryption in the encryption side and encryption in the decryption side.
- It's much stronger than double DES.

The function follows an encrypt-decrypt-encrypt (EDE) sequence.

C=E(K1,D(K2,E(K1,P)))
P=D(K1,E(K2,D(K1,C)))
By the use of triple DES with 2-key encryption, it raises the cost of the meet-in-the-middle attack to 2112. It has the drawback of requiring a key length of 56×3=16856×3=168 bits which may be somewhat unwieldy.



(b) Triple encryption

24. **Describe Elgamal digital signature**

El-gamal digital signature scheme:

- This scheme used the same keys but a different algorithm
- The algorithm creates two digital signatures, these two signatures, are used in the verification phase.
- The key generation process is the same as that of EI-gamal algorithms
- The public key remains (e1,e2,pe1,e2,p) and the private key continues to be d

Signature

- This process works as follows
- The sender selects a random number r
- The sender computes the first signature s1s1 using s1=er1modps1=e1rmodp
- The sender computes the second signature s2s2 using the equation
- s2s2 =(M−dXs1)Xr−1mod(p−1)(M−dXs1)Xr−1mod(p−1)
- Where P= large prime number
- M= original message that needs to be signed
- The sender sends M,s1,s2M,s1,s2 to the receiver.
- For eg. Let e1e1 = 10, e2e2 = 4, p=19, M=14, d=16 & r=5
- Then, s1=eR1modps1=e1Rmodp= 105mod19105mod19= 3
- s2=(M−dXs1)Xr−1mod(p−1)=(14−16X3)X5−1mod(18)s2=(M−dXs1)Xr−1mod(p−1)=(14−16X3)X5−1mod(18)= 4
- Then these signatures s1ands2s1ands2 are sent to the receiver.

Verification:

This process works as follows.

1.The receiver performs the 1st part of verification called v1v1 using the equation v1=eM1modPv1=e1MmodP
2.The receiver performs the 2nd part of verification called as v2v2 using the equation v2=es12s(s2)1modpv2=e2s1s1(s2)modp

Eg

v1=eM1modpv1=e1Mmodp = 1014mod191014mod19 = 16
and

v2=es12s(s2)1modp=43X34modpv2=e2s1s1(s2)modp=43X34modp = 5184 mod 19 = 16
Since v1v1=v2v2, the signature is valid.

25. **Explain different key distribution techniques**

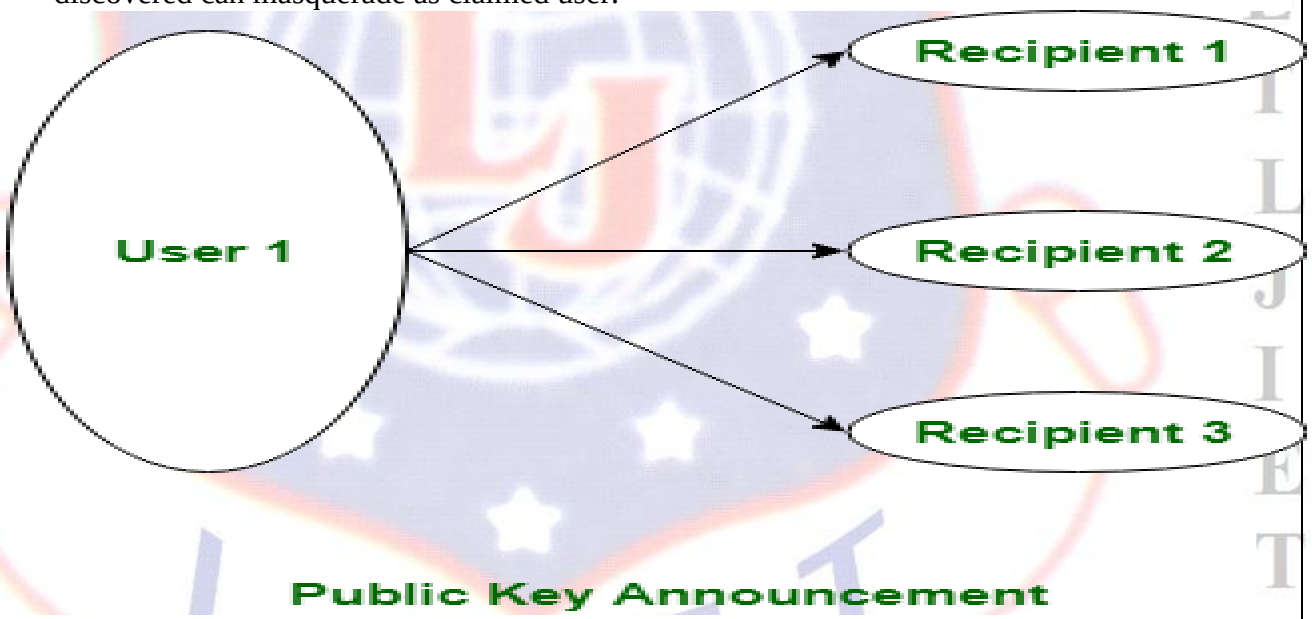
There are 2 aspects for Key Management:

1. Distribution of public keys.
2. Use of public-key encryption to distribute secret.

Distribution of Public Key:

Public key can be distributed in 4 ways: Public announcement, Publicly available directory, Public-key authority, and Public-key certificates. These are explained as following below.

1. **Public Announcement:**
   Here the public key is broadcasted to everyone. Major weakness of this method is forgery. Anyone can create a key claiming to be someone else and broadcast it. Until forgery is discovered can masquerade as claimed user.



Public Key Announcement

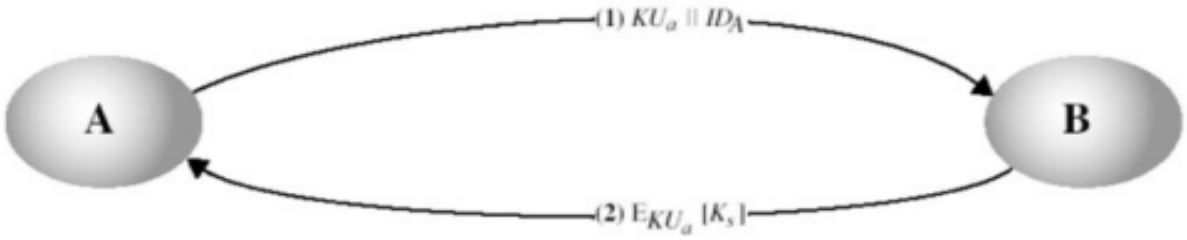
1. **Publicly Available Directory:**
   In this type, the public key is stored at a public directory. Directories are trusted here, with properties like Participant Registration, access and allow to modify values at any time, contains entries like {name, public-key}.
   Directories can be accessed electronically still vulnerable to forgery or tampering.

2. **Public Key Authority:**
   It is similar to the directory but, improve security by tightening control over distribution of keys from directory. It requires users to know public key for the directory. Whenever the keys are needed, a real-time access to directory is made by the user to obtain any desired public key securely.

3. **Public Certification:**
   This time authority provides a certificate (which binds identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied with some other info such as period of validity, rights of use etc. All of this content is signed by the trusted Public-Key or Certificate Authority (CA) and it can be verified by anyone possessing the authority's public-key.

Simple Secret-Key Distribution

An extremely simple scheme put forward by Ralph Merkle is illustrated in figure 9.4.
If A wishes to communicate with B, the following procedure is employed:
1. A generates a public/private key pair {KUA, KRA} and transmits a message to B consisting of KUA, and an identifier of A, IDA.
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
3. A computes DKRA[EKUA[Ks]] to recover the secret key. Since only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards KUA and KRA and B discards KUA.



- A and B can now securely communicate encryption and the session keys Ks. At the completion of the exchange, both A and B discard Ks. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping.
- The protocol is vulnerable to an active attack. If an opponent, E, has control of the intervening communications channel, then he can compromise the communications in the following way without being detected:

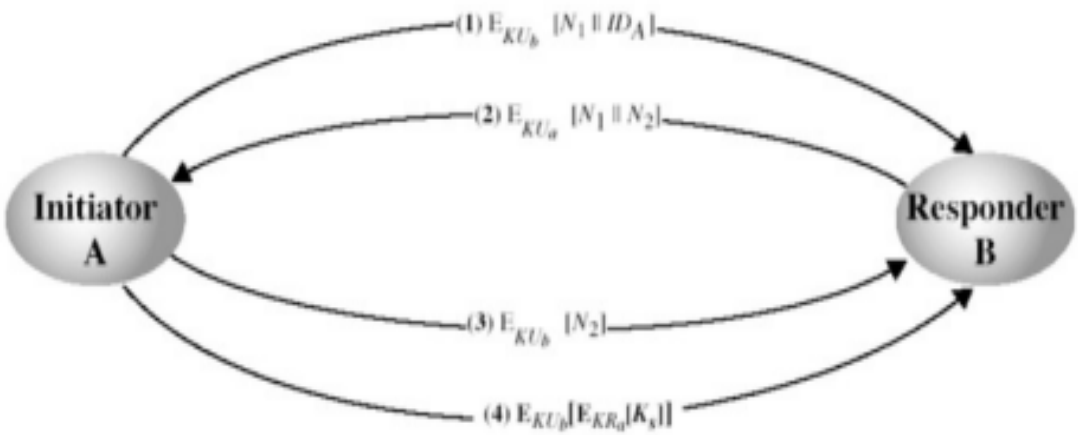
1. A generates a public/private key pair {KUA, KRA} and transmits a message intended for B consisting of KUA and identifier of A, IDA.
2. E intercepts the message, creates its own public/private key pair {KUE, KRE} and transmits KUE||IDA to B.
3. B generates a secret key, Ks, and transmits EKUE[Ks].
4. E intercepts the message, and learns Ks by computing DKRE [EKUE[Ks]].
5. E transmits EKUA[Ks] to A.

The result is that both A and B know Ks and are unaware that Ks has also been revealed to E. A and B can now exchange messages using Ks. E no longer actively interferes with the communications channel but simply eavesdrops. Knowing, Ks, E can decrypt all messages, and both A and B are unaware of the problem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping

Secret Key Distribution with Confidentiality and Authentication

Figure 9.5 provides protection against both active and passive attacks. For this example it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section. Then the following steps occur:
1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to uniquely identify this transaction.
2. B sends a message to A encrypted with KUA and containing A's nonce (N1) as well as a new nonce generated by B (N2). Since only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondentis B.
3. A returns N2, encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key Ks and sends M = EKUb[EKRA[Ks]] to B. Encryption of this message with B's public key ensures that only B can read it, encryption with A's private key ensures that only A could have sent it.

5. B computes DKUA[DKRb[M]] to recover the secret key.
This scheme ensures both confidentiality and authentication in the exchange of a secret Key



A Hybrid Scheme

- Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes. This scheme retains the use of a key distribution
- centre (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key. A public key scheme is used to distribute the
- master keys