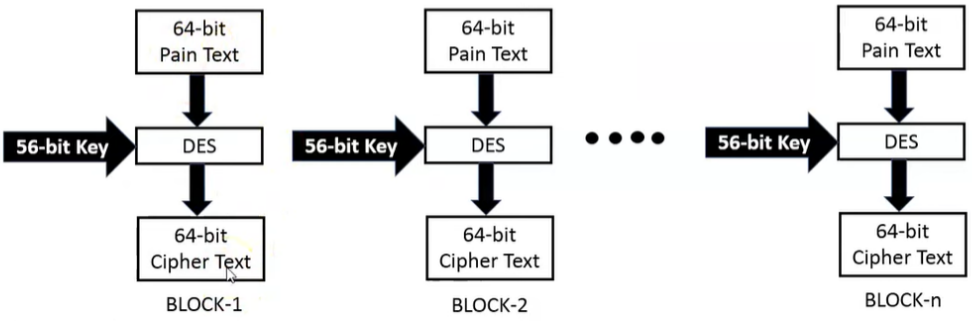
Block Ciphers & Public Key Cryptography

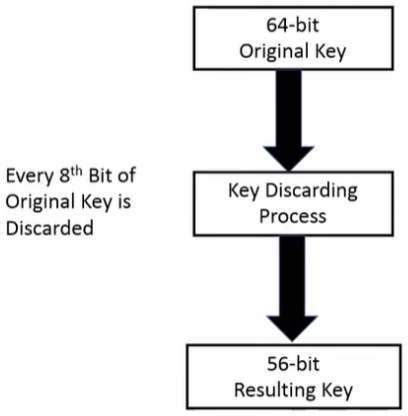
DES (Data Encryption Standard)

- **Introduction**
  - Developed in early 1970's at IBM and submitted to NBS .
  - DES is landmark in cryptographic algorithms.
  - DES works based on Feistel Cipher Structure.
  - DES is symmetric cipher algorithm and use block cipher method for encryption and decryption..

- **Figure shows process of DES**



- **Key discarding process**

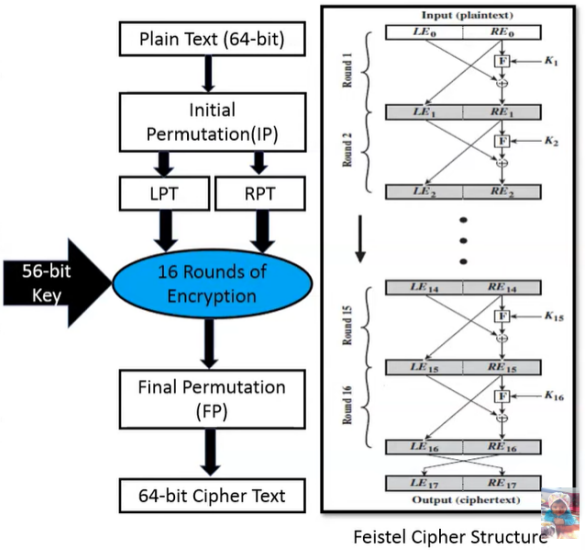


1	2	21	38	58	15	37	26
22	55	44	3	53	27	11	60
49	28	14	42	61	48	63	41
18	39	56	10	64	16	62	8
45	40	20	54	4	33	34	52
7	30	47	59	32	5	35	25
29	12	13	6	24	46	57	36
17	23	50	31	43	51	9	19

1	2	21	38	58	15	37
22	55	44	3	53	27	11
49	28	14	42	61	48	63
18	39	56	10	64	16	62
45	40	20	54	4	33	34
7	30	47	59	32	5	35
29	12	13	6	24	46	57
17	23	50	31	43	51	9

❖ Steps of DES

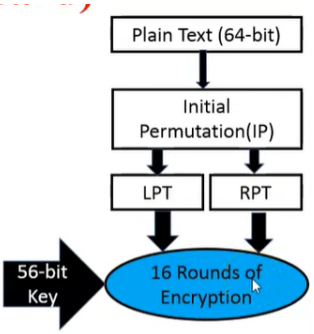
1. 64-bit plain text block is given to Initial Permutation (IP) function.
2. IP performed on 64-bit plain text block.
3. IP produced two halves of the permuted block known as Left Plain Text (LPT) and Right Plain Text (RPT).
4. Each LPT and RPT performed 16-rounds of encryption process.
5. LPT and RPT rejoined and Final Permutation (FP) is performed on combined block.
6. 64-bit Cipher text block is generated.



Feistel Cipher Structure

Initial Permutation (IP) & Generate LPT -RPT

- Initial Permutation performed only once.
- Bit sequence have changed as per IP table.
- **For Example:**
  - ✓ 1st bit take 40th Position
  - ✓ 58th bit take 1st position

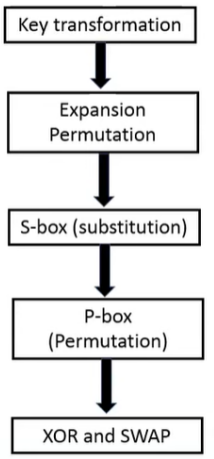


58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

- Out put of IP is divided into two equal halves known as LPT, RPT. (LPT – 32 bit, RPT – 32 bit)

16 Rounds of Encryption

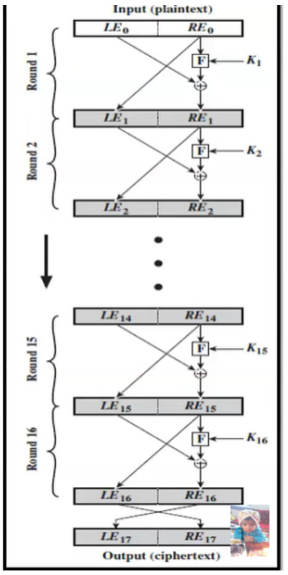
1. Key Transformation (56-bit key)
   - Key Bit Shifted per round
   - Compression Permutation
2. Expansion permutation of Plain Text and X-OR (P.T. size: 48 bit, C.T. size: 48 bit)
3. S-box Substitution
4. P-box (Permutation)
5. X-OR and Swap.



Key Bit Shifted per Round

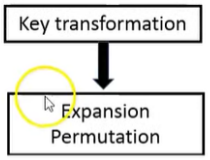
- 56-bit key is divided into two halves each of 28-bits
- Circular left shift is performed on each halves
- Shifting of Bit position is depending on round
  - For ***round number*** 1,2,9 and 16 shift is done by one position

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Key bit shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1



Compression Permutation

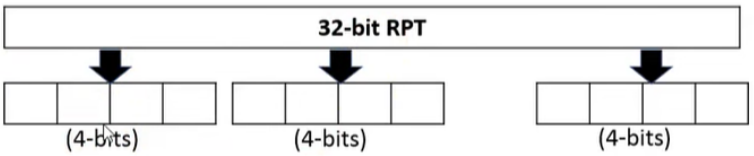
- 56-bit input with bit shifting position
- Generates 48-bit key (Compression of Key bit)
- Drop 9, 18, 22, 25, 35, 38, 43 and 54 bits.



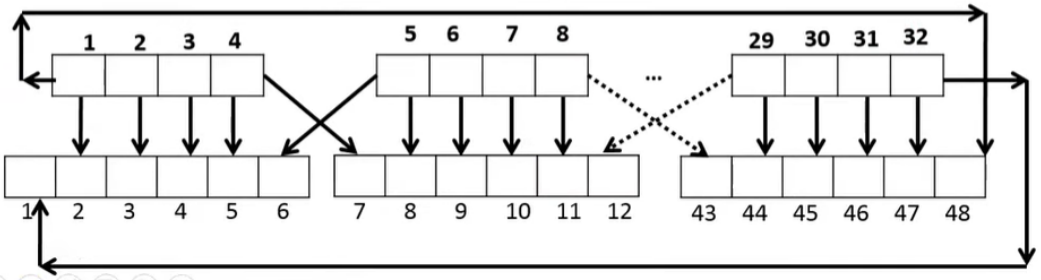
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Expansion Permutation

- 32-bit RPT of IP is expanded to 48-bits
- Expansion permutation steps:
  - 32-bit RPT is divided into 8-blocks each of 4-bits

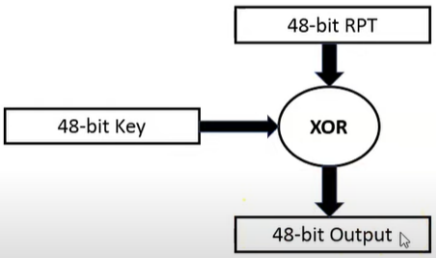


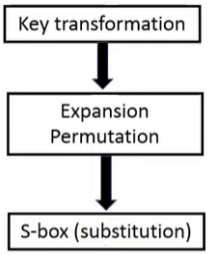
- Each 4-bit block is expanded to 6-bit and produce 48-bit output



Expansion Permutation

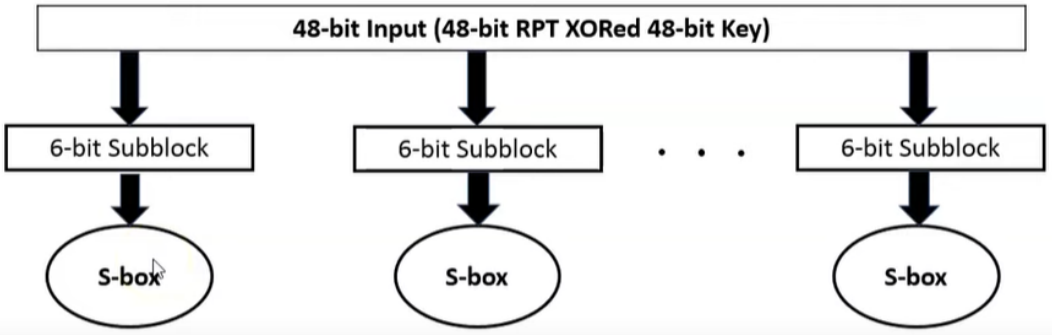
- 48-bit RPT is XORed with 48-bit Key and output is given to S-Box.



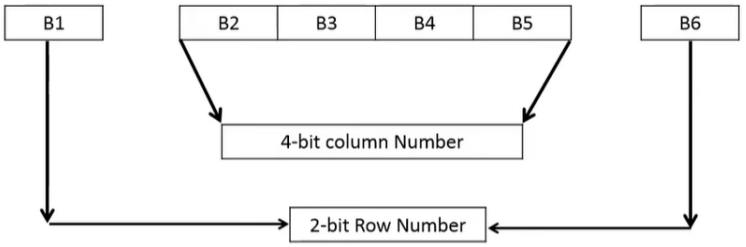


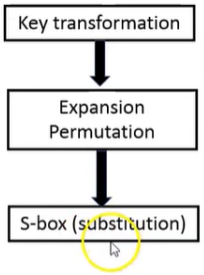
- Now 48-bit XORed output convert into 32 bit.

S-BOX Substitution



S-BOX Working





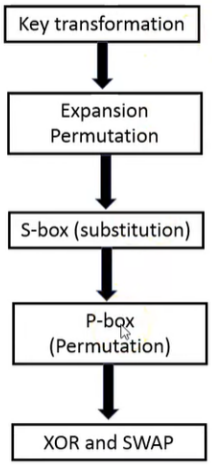
S_5		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Example: 011011 → 1001

P-BOX Permutation

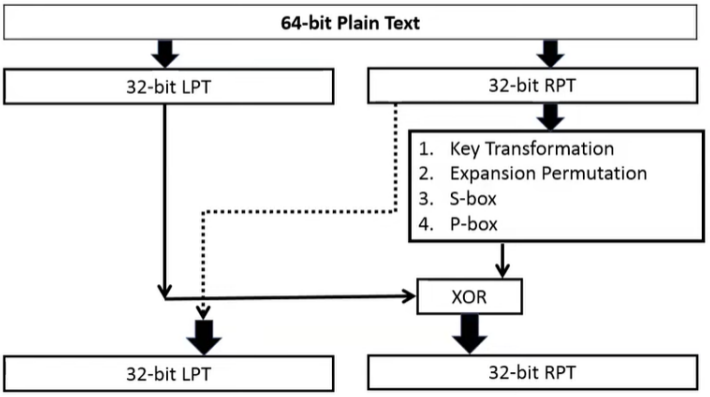
- Output of s-box is given to p-box
- 32-bit is permuted with 16 x 2 permutation table
- **For Example:**
  - ✓ 16th bit of S-box take 1st Position as per below permutation table.

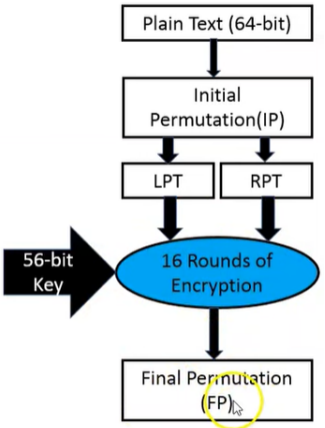
P – Box Table															
16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25



XOR and SWAP

- 32-bit LPT is XORed with 32-bit p-box.





- 1st round of encryption is completed. Now remaining 15 rounds will be performed same as 1st round.

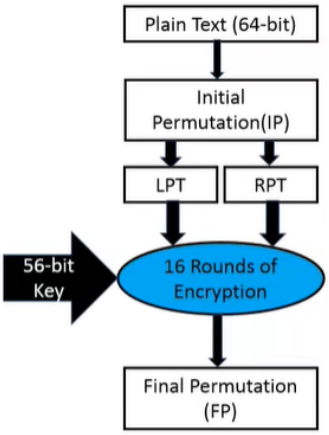
Final Permutation

- At the end of the 16 rounds, the final permutation is performed (only once).

- **For Example:**
  - ✓ 40th bit of input take 1st Position as per below permutation table.

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

- The output of the final permutation is *the 64-bit encrypted block (64-bit cipher text block).*

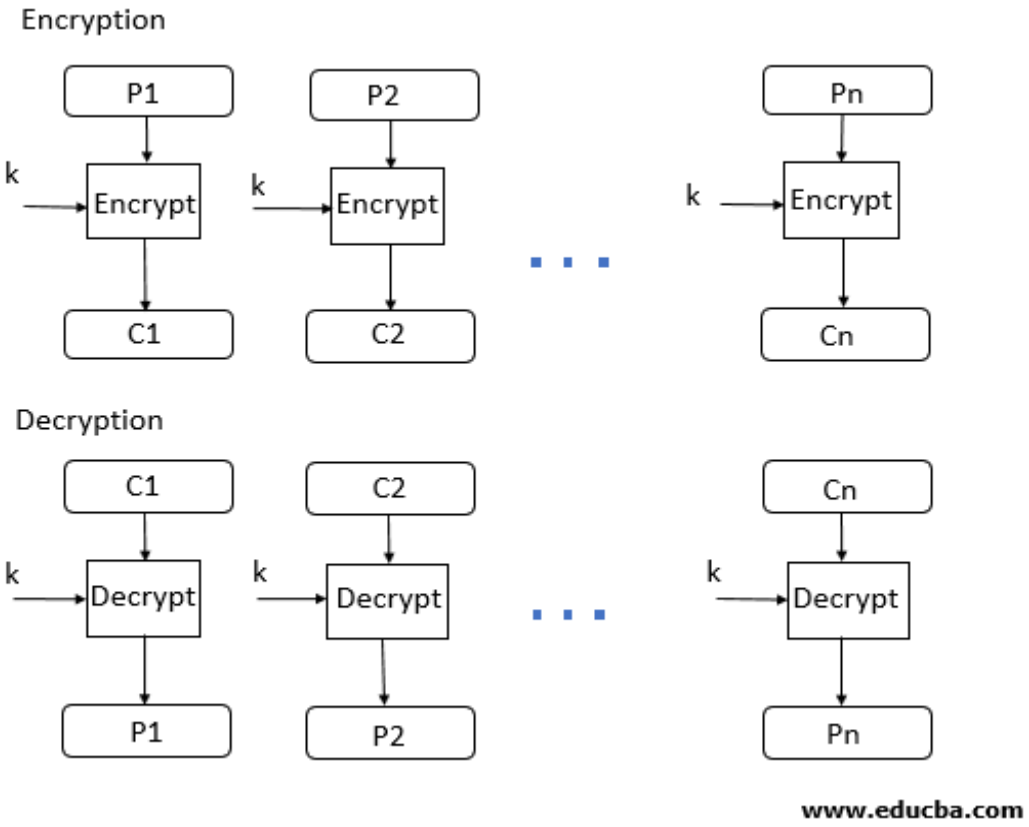


Introduction to Block Cipher modes

There are five types of operations in block cipher modes, ECB (Electronic Code Block) mode, CBC (Cipher Block Chaining) mode, CFB (Cipher Feedback) mode, OFB (Output Feedback) mode and CTR (Counter) mode. Where ECB and CBC mode works on block ciphers, and CFB and OFB mode works on block ciphers acting as stream ciphers. ECB is used for transmitting a single value insecure manner, CBC is used for encrypting blocks of text authentication, CFB is used for transmitting an encrypted stream of data authentication, OFB is used for transmitting an encrypted stream of data, CTR is used for transmitting block-oriented applications.

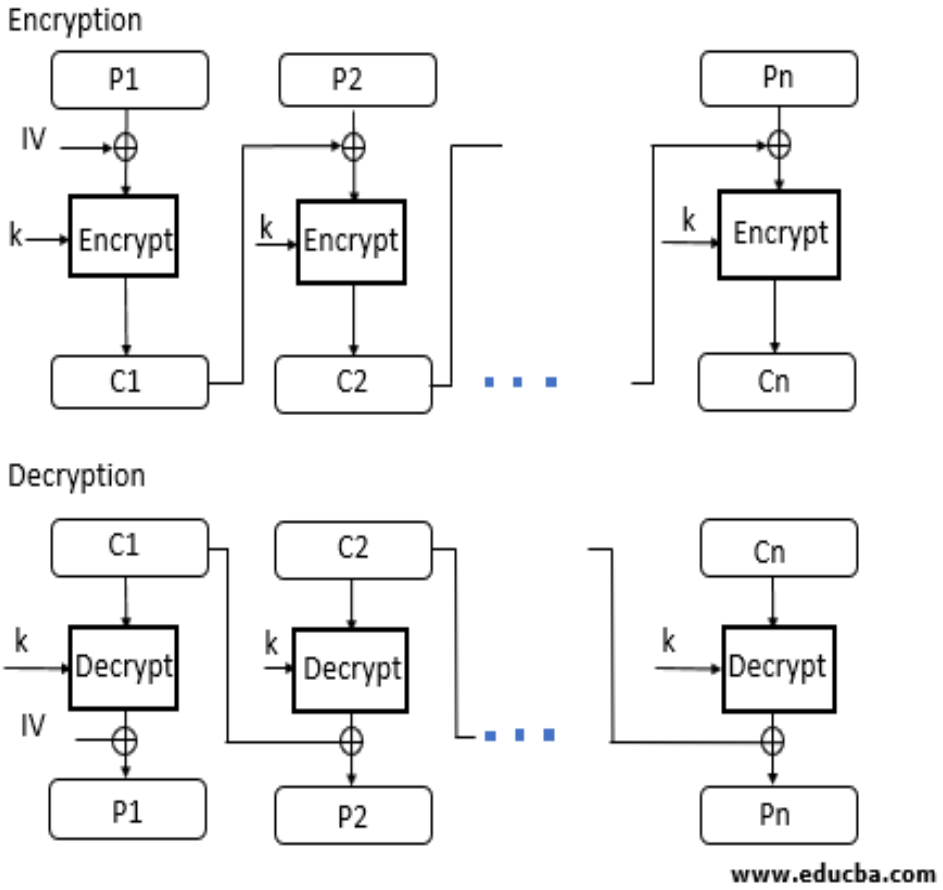
1. ECB mode

- ECB mode stands for **Electronic Code Block Mode**. It is one of the simplest modes of operation. In this mode, the plain text is divided into a block where each block is 64 bits. Then each block is encrypted separately. The same key is used for the encryption of all blocks. Each block is encrypted using the key and makes the block of ciphertext.
- At the receiver side, the data is divided into a block, each of 64 bits. The same key which is used for encryption is used for decryption. It takes the 64-bit ciphertext and, by using the key convert the ciphertext into plain text.
- As the same key is used for all blocks' encryption, if the block of plain text is repeated in the original message, then the ciphertext's corresponding block will also repeat. As the same key used for tor all block, to avoid the repetition of block ECB mode is used for an only small message where the repetition of the plain text block is less.



2. CBC Mode

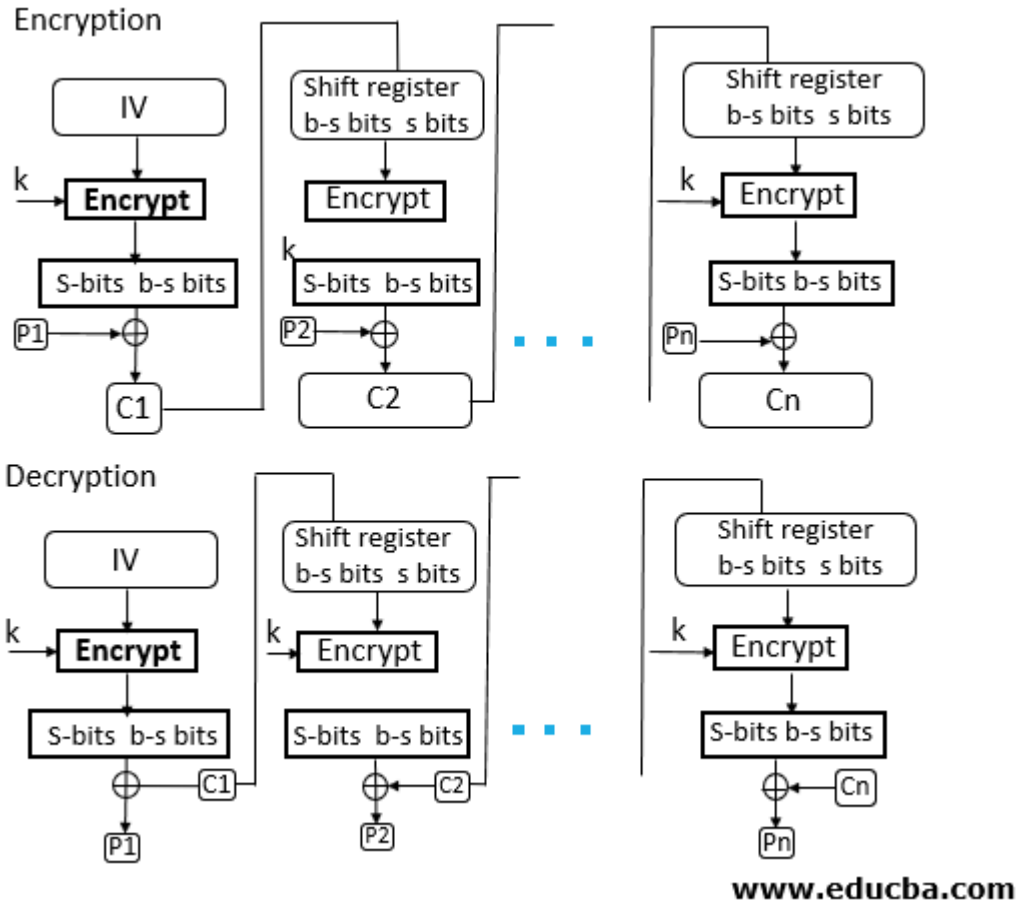
- In CBC, the previous cipher block is given as input to the next encryption algorithm after XOR with the original plaintext block. In a nutshell here, a cipher block is produced by encrypting an XOR output of the previous cipher block and present plaintext block.
- CBC Mode ensures that if the block of plain text is repeated in the original message, it will produce a different ciphertext for corresponding blocks. Note that the key which is used in CBC mode is the same; only the IV is different, which is initialized at a starting point.



3. CFB Mode

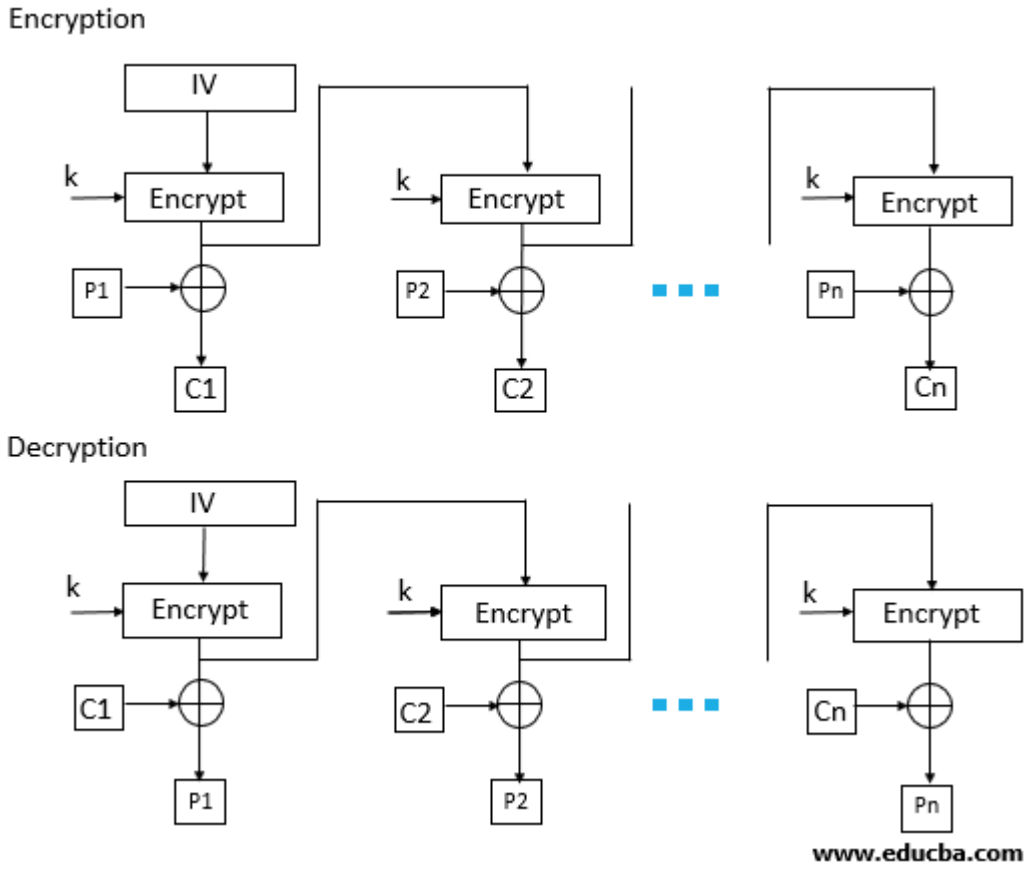
4.

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 *s* and *b-s* bits.The left-hand side *s* bits are selected along with plaintext bits to which an XOR operation is applied. The result is given as input to a shift register having b-s bits to lhs,s bits to rhs and the process continues. The encryption and decryption process for the same is shown below, both of them use encryption algorithms.



4. OFB mode

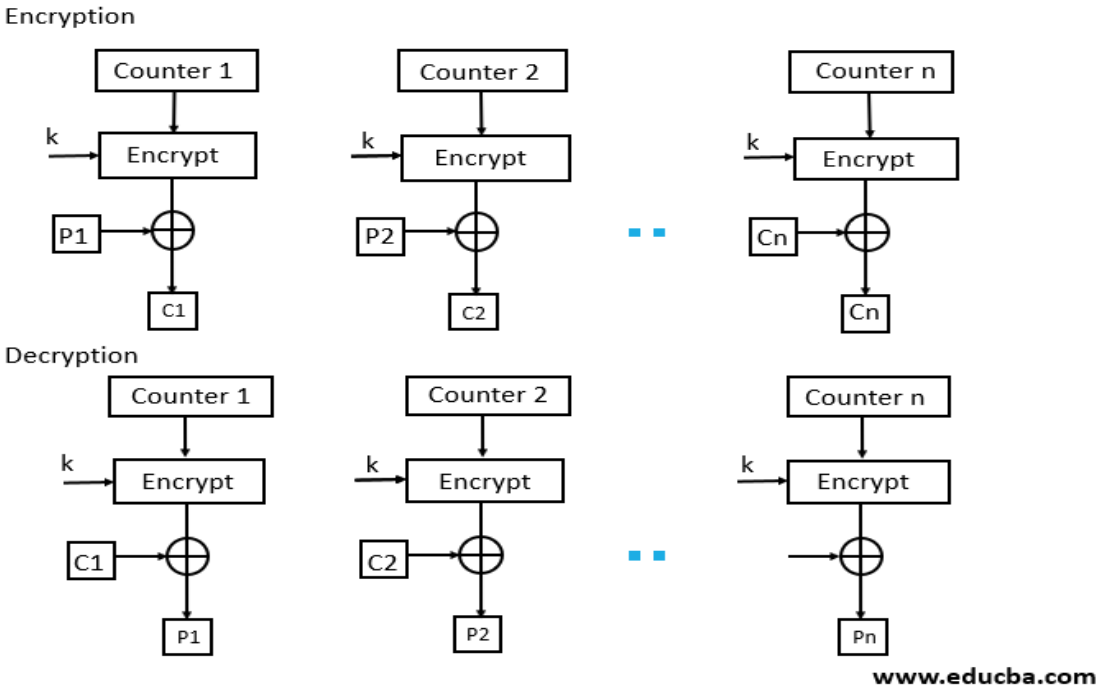
- OFB Mode stands for output feedback Mode. OFB mode is similar to CDB mode; the only difference is in CFB, the ciphertext is used for the next stage of the encryption process, whereas in OFB, the output of the IV encryption is used for the next stage of the encryption process.
- The IV is encrypted using the key and form encrypted IV. Plain text and leftmost 8 bits of encrypted IV are combined using XOR and produce the ciphertext.
- For the next stage, the ciphertext, which is the form in the previous stage, is used as an IV for the next iteration. The same procedure is followed for all blocks.



5. CTR Mode

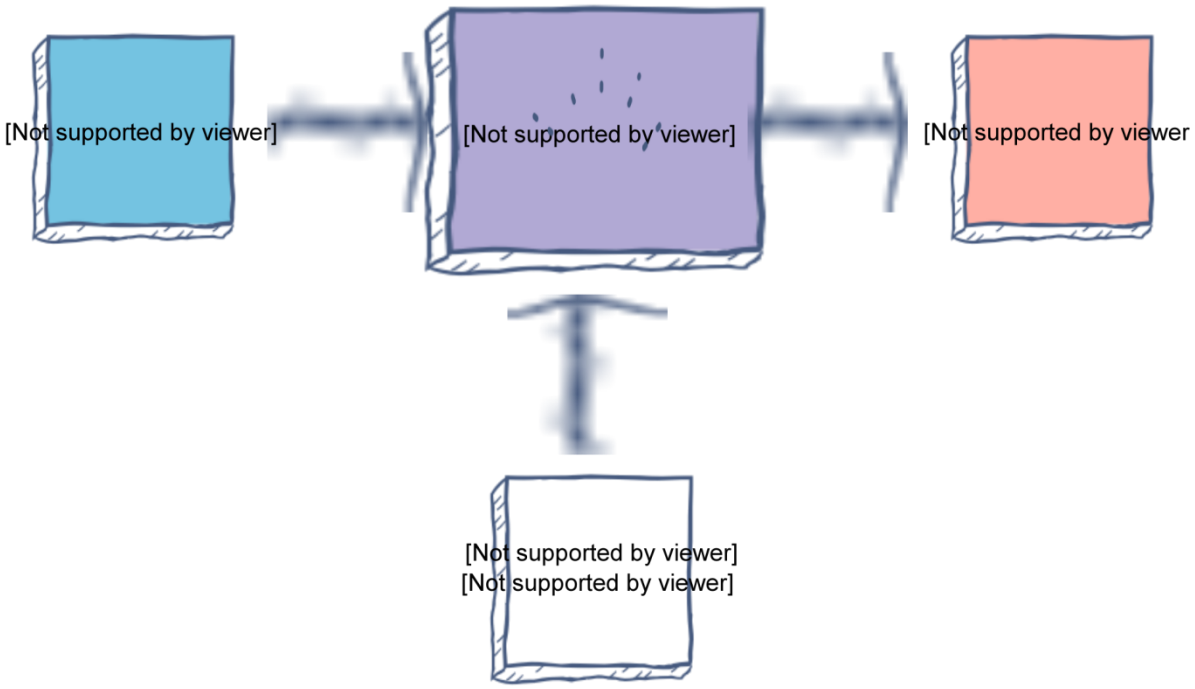
- CTR Mode stands for counter mode. As the name is counter, it uses the sequence of numbers as an input for the algorithm. When the block is encrypted, to fill the next register next counter value is used.
  Note: the counter value will be incremented by 1.
- For encryption, the first counter is encrypted using a key, and then the plain text is XOR with the encrypted result to form the ciphertext.
- The counter will be incremented by 1 for the next stage, and the same procedure will be followed for all blocks. For decryption, the same sequence will be used. Here to convert ciphertext into plain text, each ciphertext is XOR with the encrypted counter. For the next stage, the counter will be

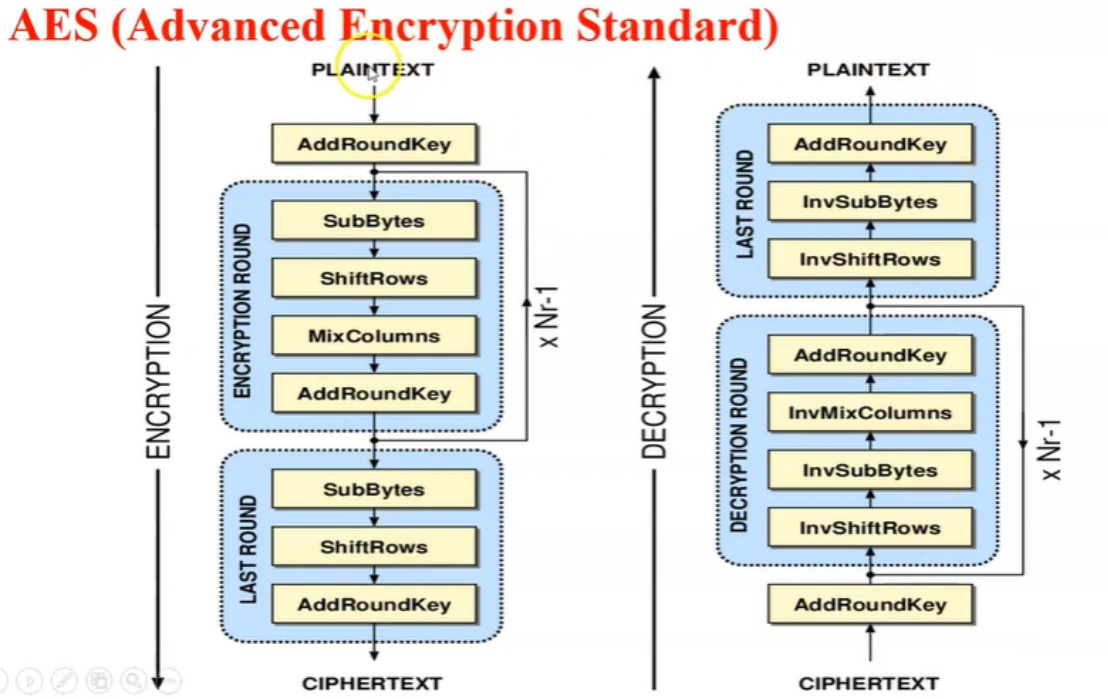
incremented by the same will be repeated for all Ciphertext blocks.



What is the AES algorithm?

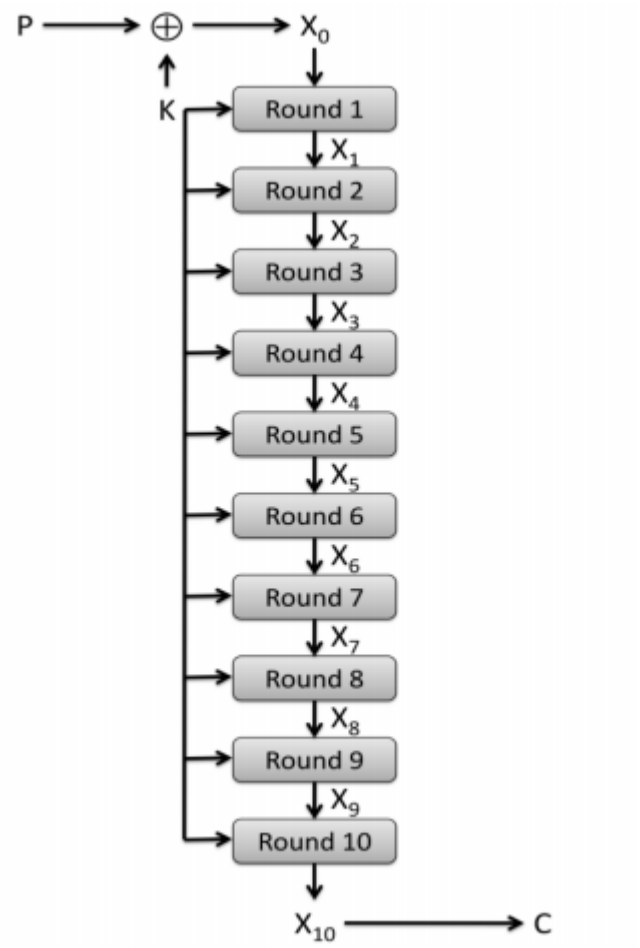
The **AES algorithm** (also known as the **Rijndael algorithm**) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard.





How does AES work?

The AES algorithm uses a substitution-permutation, or SP network, with multiple rounds to produce ciphertext. The number of rounds depends on the key size being used. A 128-bit key size dictates ten rounds, a 192-bit key size dictates 12 rounds, and a 256-bit key size has 14 rounds. Each of these rounds requires a round key, but since only one key is inputted into the algorithm, this key needs to be expanded to get keys for each round, including round 0.



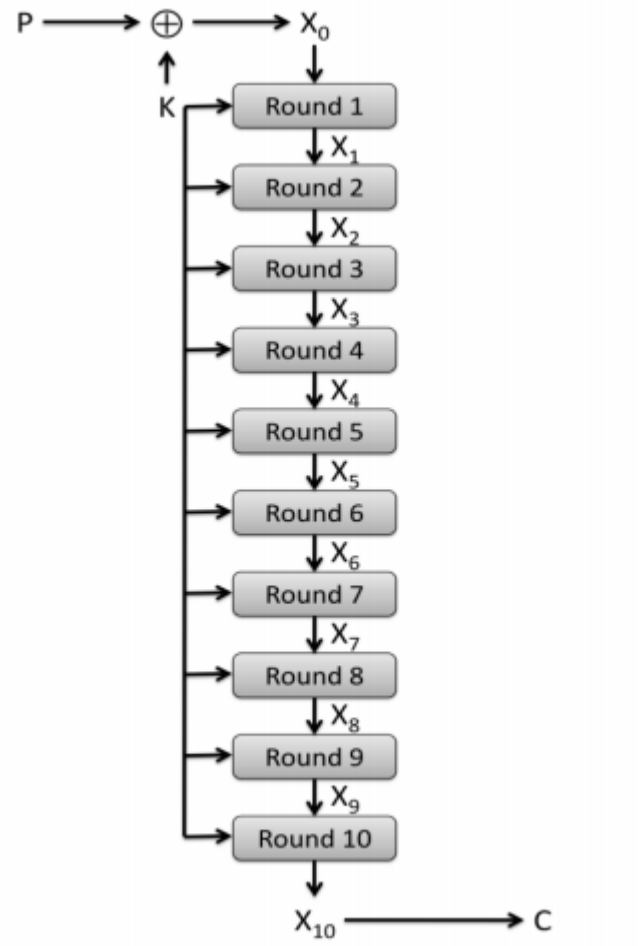
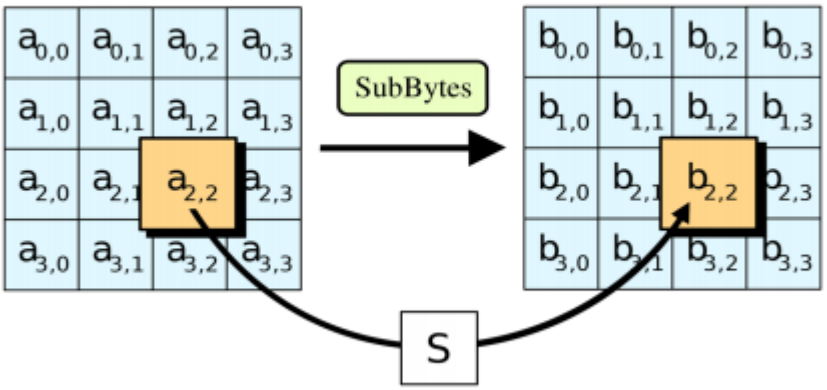


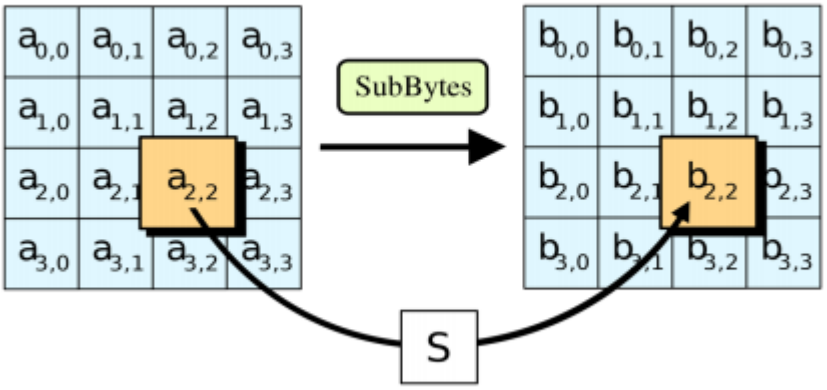
Figure 2

Steps in each round

Each round in the algorithm consists of four steps.

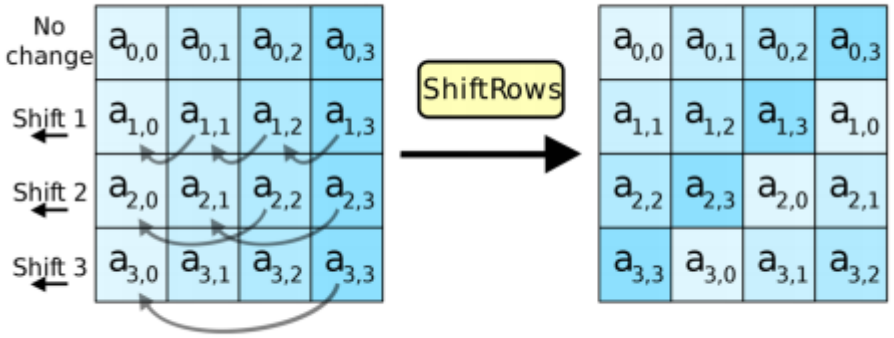
1. Substitution of the bytes
In the first step, the bytes of the block text are substituted based on rules dictated by predefined S-boxes (short for substitution boxes).

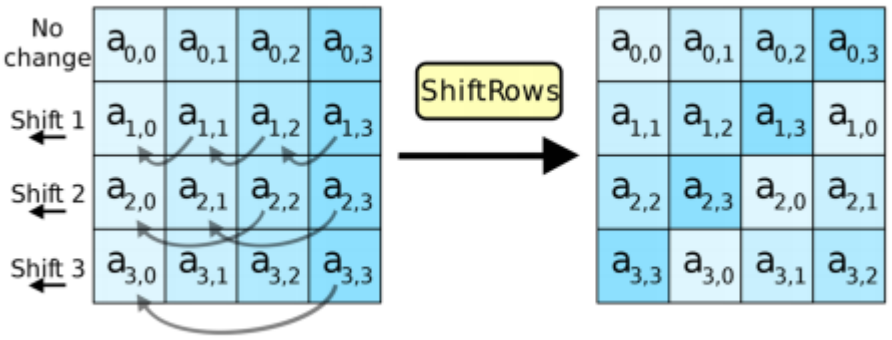




2. Shifting the rows

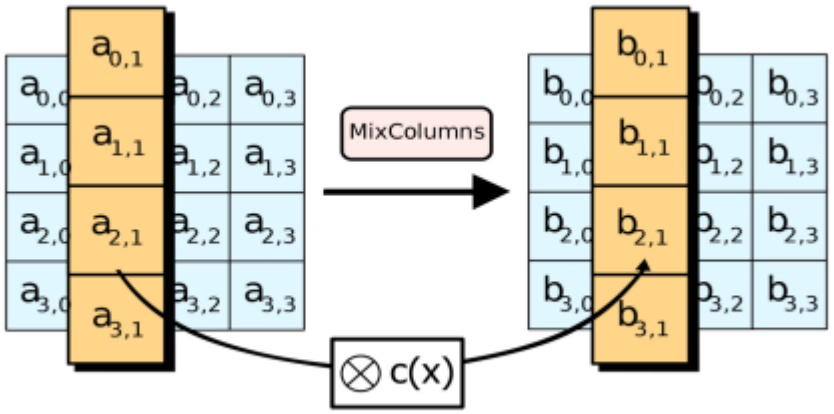
Next comes the permutation step. In this step, all rows except the first are shifted by one, as shown below.

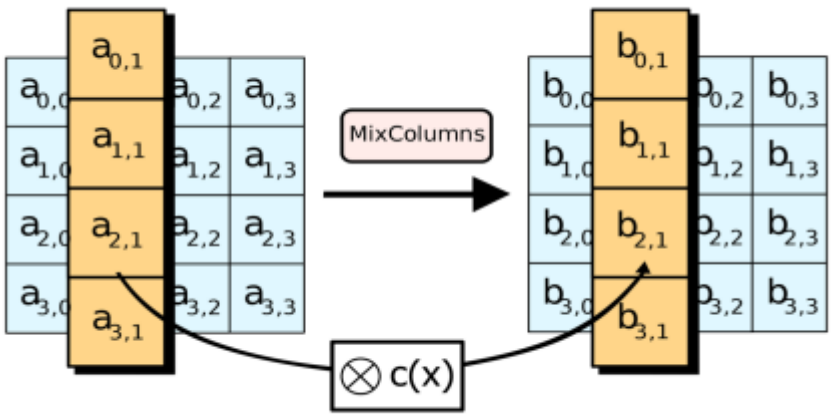




3. Mixing the columns

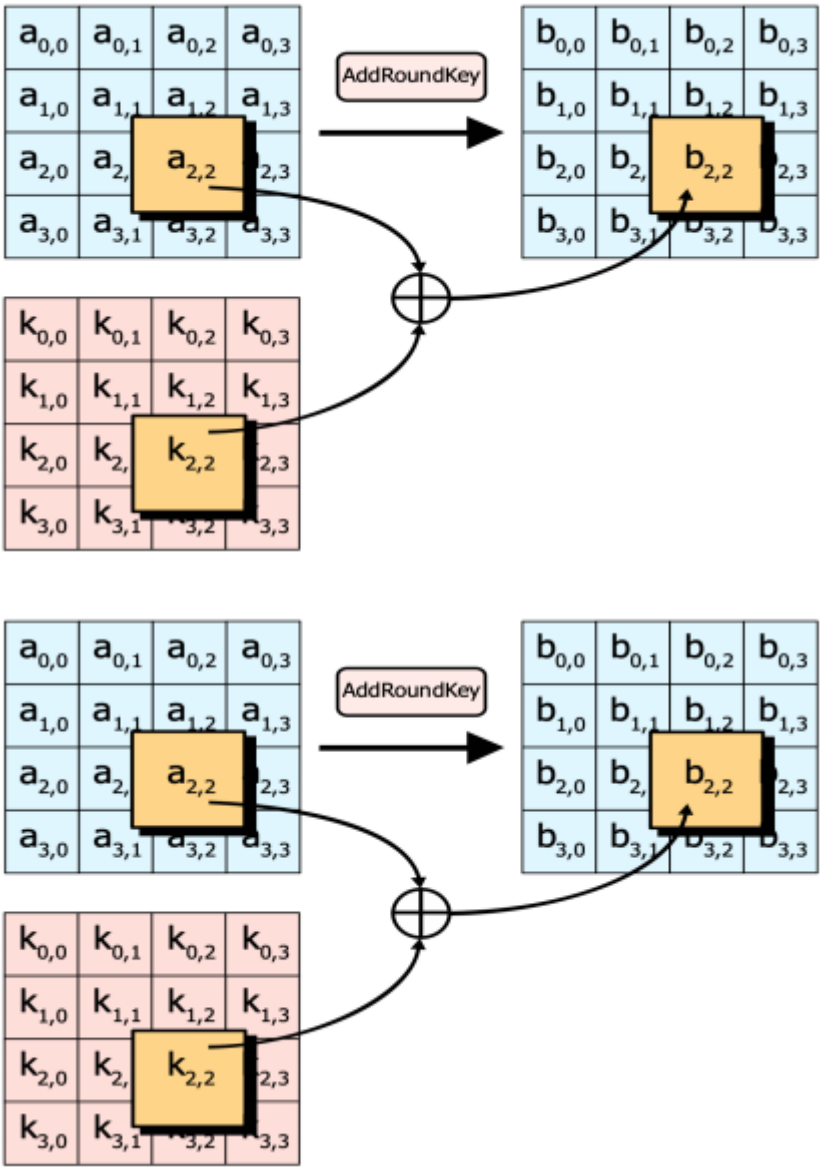
In the third step, the Hill cipher is used to jumble up the message more by mixing the block's columns.





4. Adding the round key

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



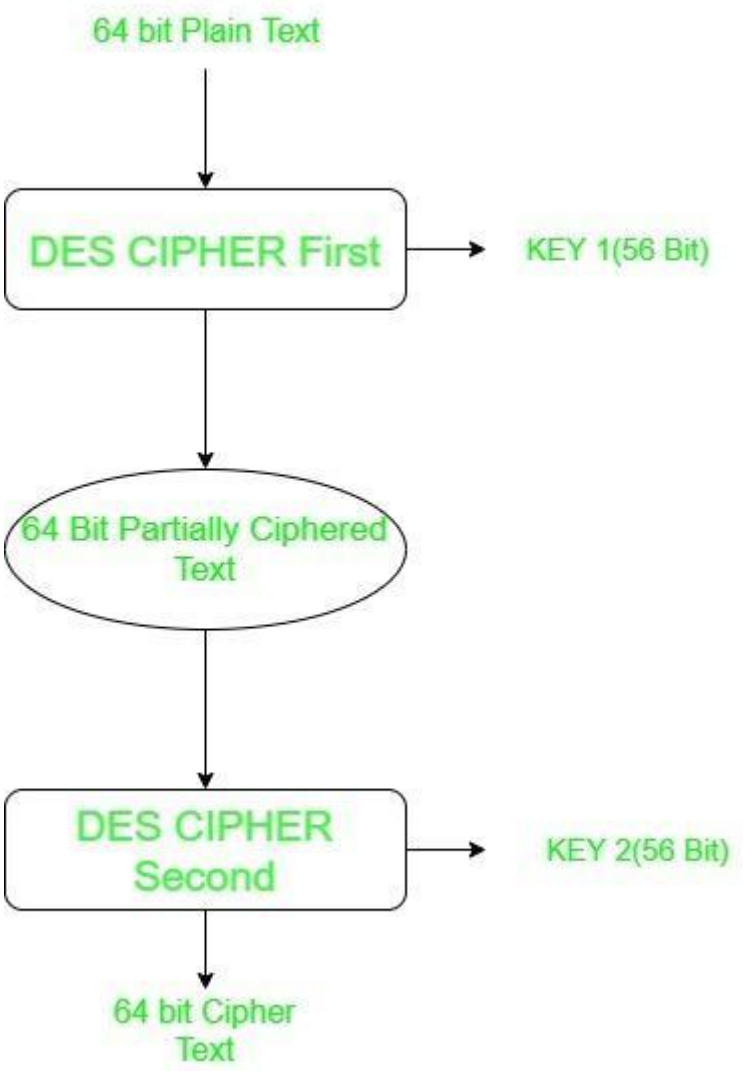
When done repeatedly, these steps ensure that the final ciphertext is secure.

As we know the Data encryption standard (DES) uses 56 bit key to encrypt any plain text which can be easily be cracked by using modern technologies. To prevent this from happening double DES and triple DES were introduced which are much more secured than the original DES because it uses 112 and 168 bit keys respectively. They offer much more security than DES.

Double DES:

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

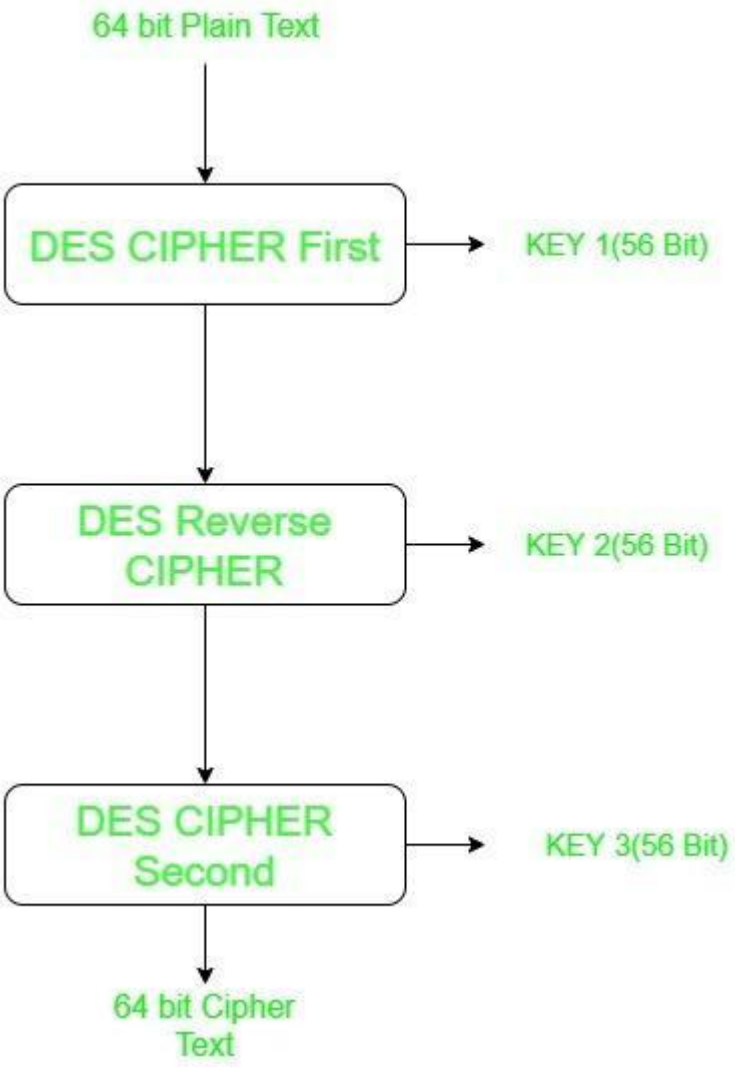
goes into first DES instance which then converted into a 64 bit middle text using the first key and then it goes to second DES instance which gives 64 bit cipher text by using second key.



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.

Triple DES:

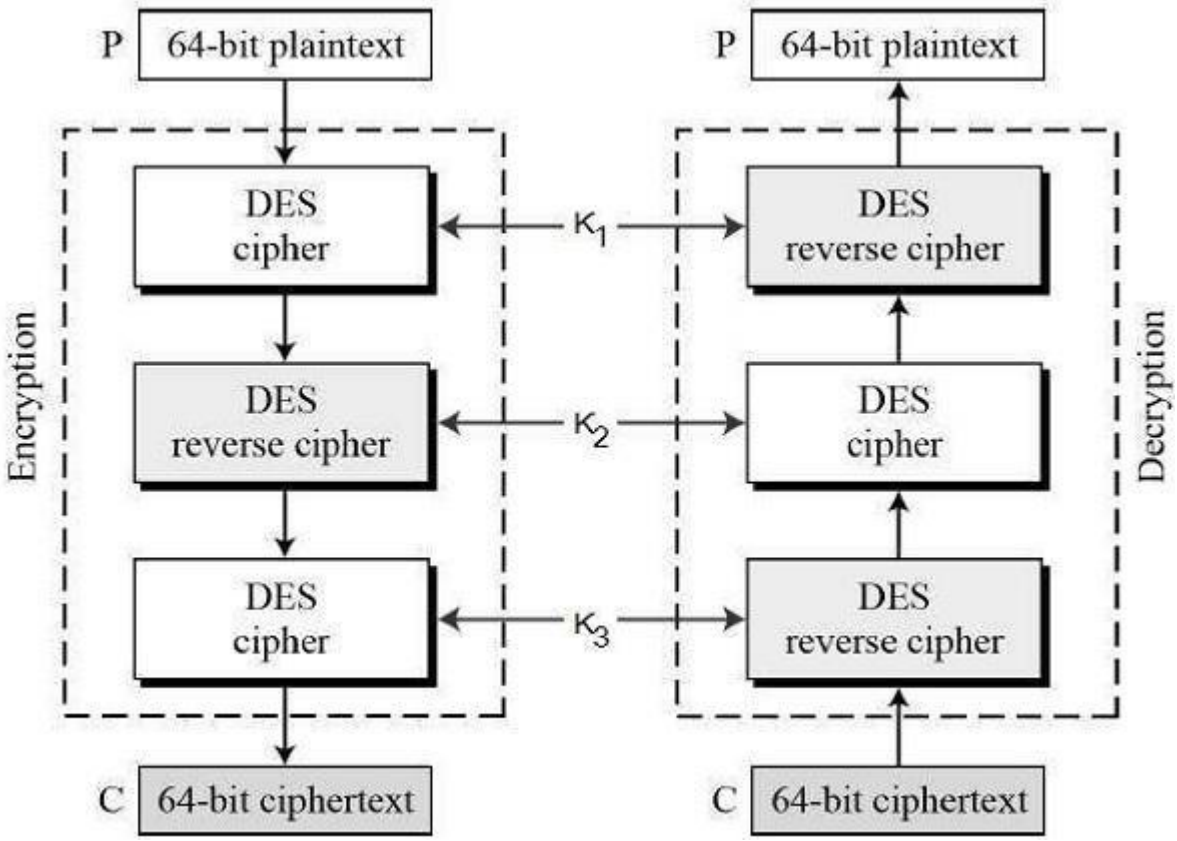
Triple DES is a encryption technique which uses three instance of DES on same plain text. It uses there different types of key choosing technique in first all used keys are different and in second two keys are same and one is different and in third all keys are same.



Triple DES is also vulnerable to meet-in-the middle attack because of which it give total security level of 2^112 instead of using 168 bit of key. The block collision attack can also be done because of short block size and using same key to encrypt large size of text. It is also vulnerable to sweet32 attack.

3-KEY Triple DES

Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows −



The encryption-decryption process is as follows −

- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2 and finally decrypt with K_1.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K_1, K_2 and K_3 to be the same value. This provides backwards compatibility with DES.

-Blowfish

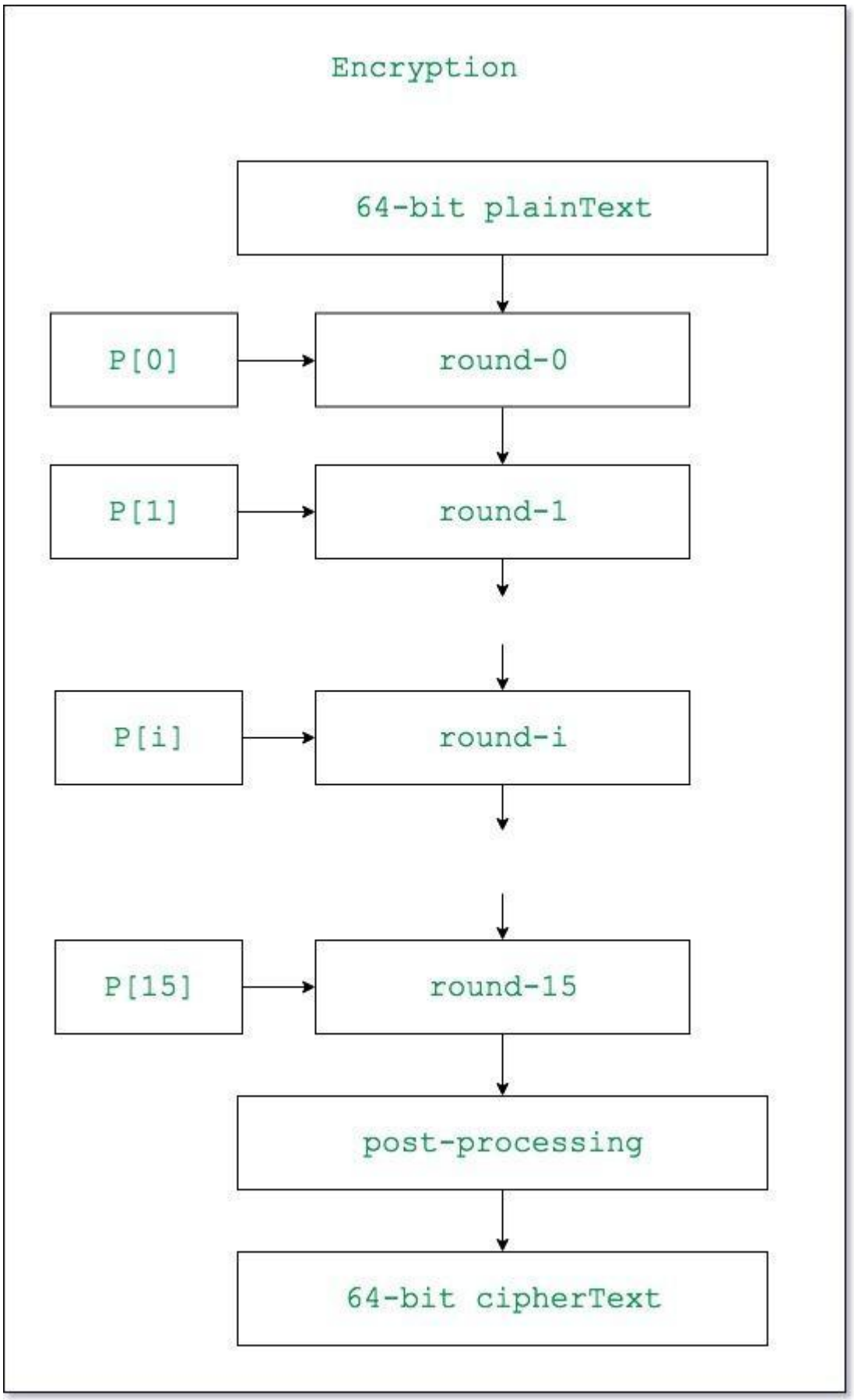
Blowfish is an encryption technique designed by **Bruce Schneier** in 1993 as an alternative to DES Encryption Technique. It is significantly faster than DES

and provides a good encryption rate with no effective cryptanalysis technique found to date. It is one of the first, secure block cyphers not subject to any patents and hence freely available for anyone to use.

1. **blockSize**: 64-bits
2. **keySize**: 32-bits to 448-bits variable size
3. **number of subkeys**: 18 [P-array]
4. **number of rounds**: 16
5. **number of substitution boxes**: 4 [each having 512 entries of 32-bits each]

Blowfish Encryption Algorithm

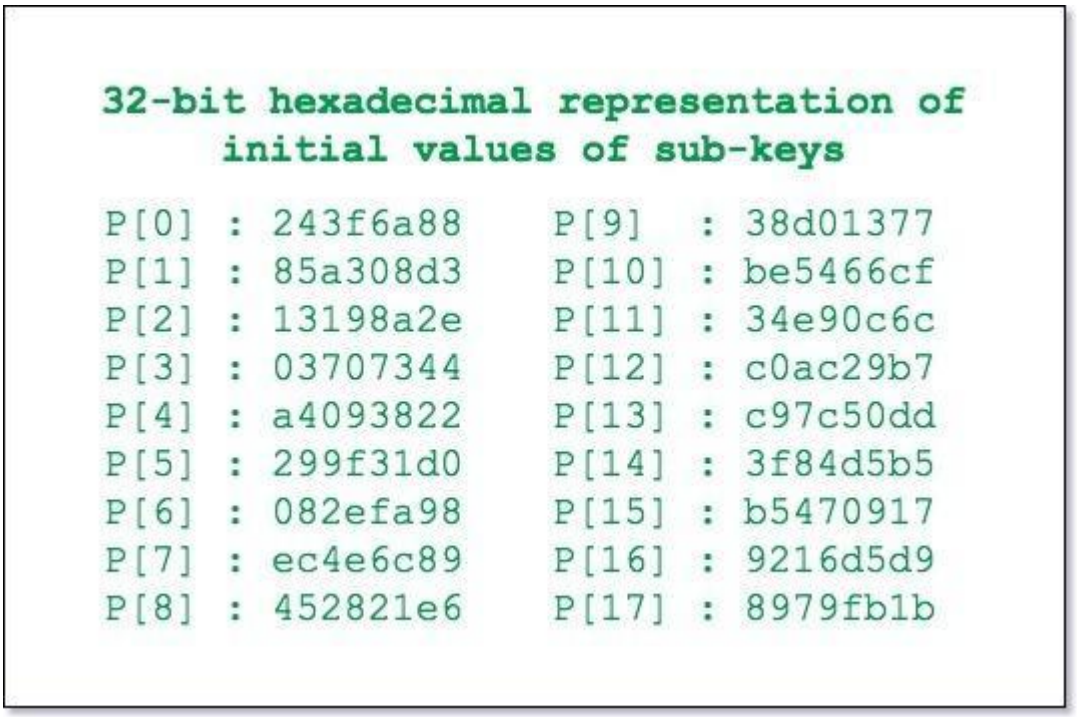
The entire encryption process can be elaborated as:


Encryption
64-bit plainText
P[0]
round-0
P[1]
round-1
P[i]
round-i
P[15]
round-15
post-processing
64-bit cipherText

Step1: Generation of subkeys:

- 18 subkeys{P[0]…P[17]} are needed in both encryption as well as decryption process and the same subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

```
P[0] = "243f6a88"
P[1] = "85a308d3"
.
.
.
P[17] = "8979fb1b"
```



- Now each of the subkey is changed with respect to the input key as:

```
P[0] = P[0] xor 1st 32-bits of input key
P[1] = P[1] xor 2nd 32-bits of input key
.
.
.
P[i] = P[i] xor (i+1)th 32-bits of input key
```

```
(roll over to 1st 32-bits depending on the key
length)
.
.
.
P[17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)
```

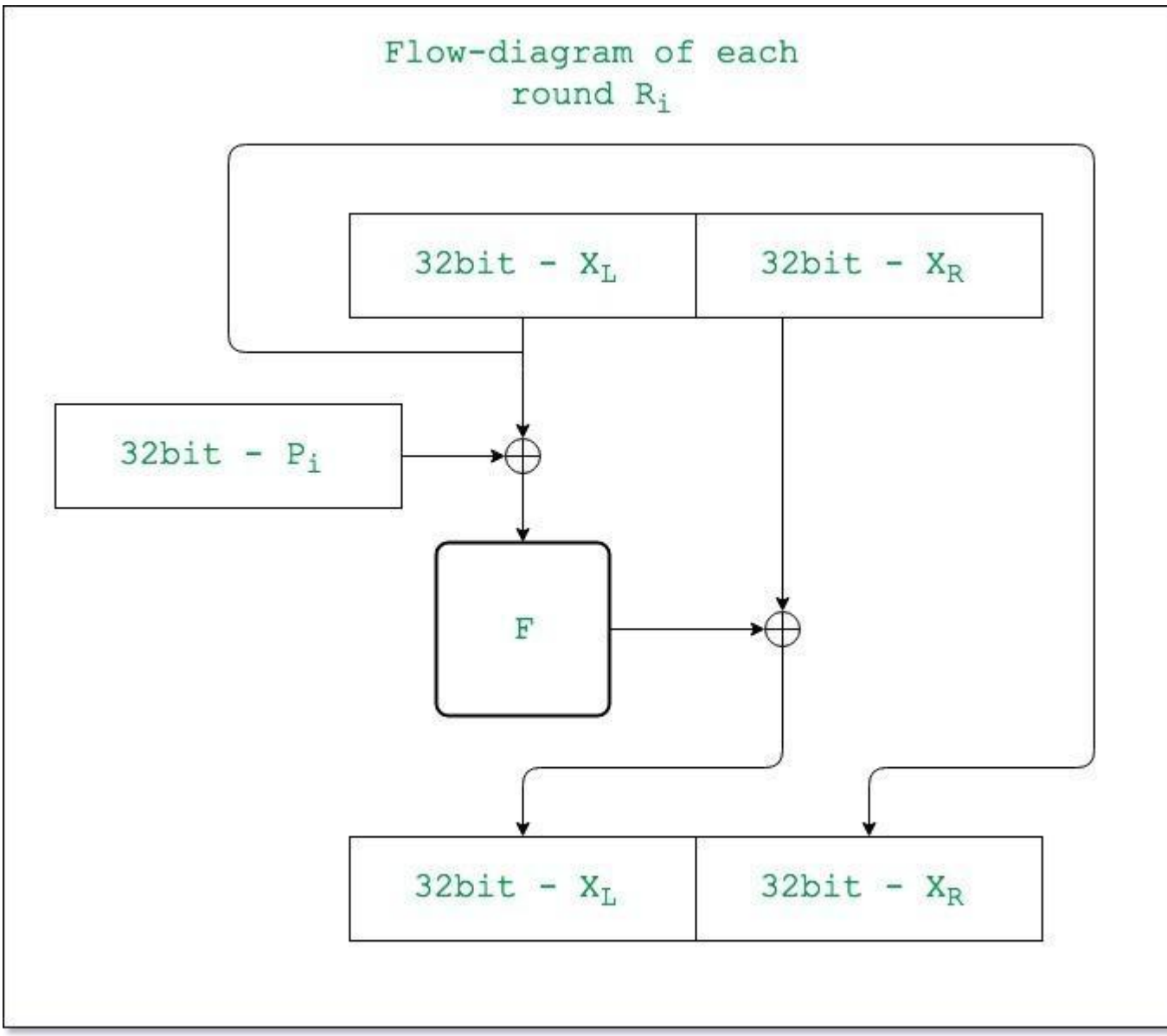
The resultant P-array holds 18 subkeys that is used during the entire encryption process

Step2: initialise Substitution Boxes:

- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption aswell as decryption process with each S-box having 256 entries{S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.
- It is initialized with the digits of pi(?) after initializing the P-array. You may find the s-boxes in here!

Step3: Encryption:

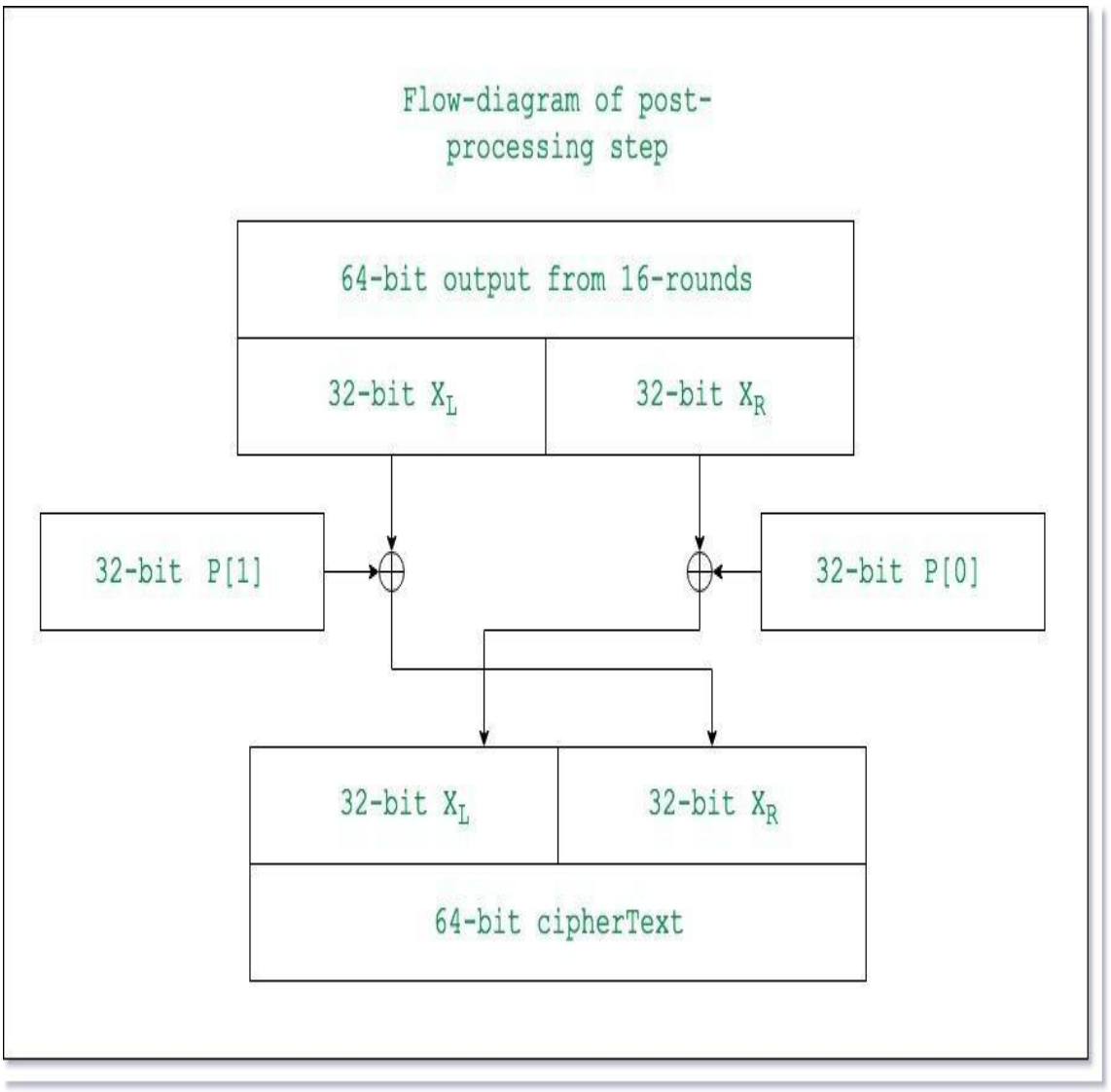
- The encryption function consists of two parts:
  a. Rounds: The encryption consists of 16 rounds with each round(Ri) taking inputs the plainText(P.T.) from previous round and corresponding subkey(Pi). The description of each round is as follows:

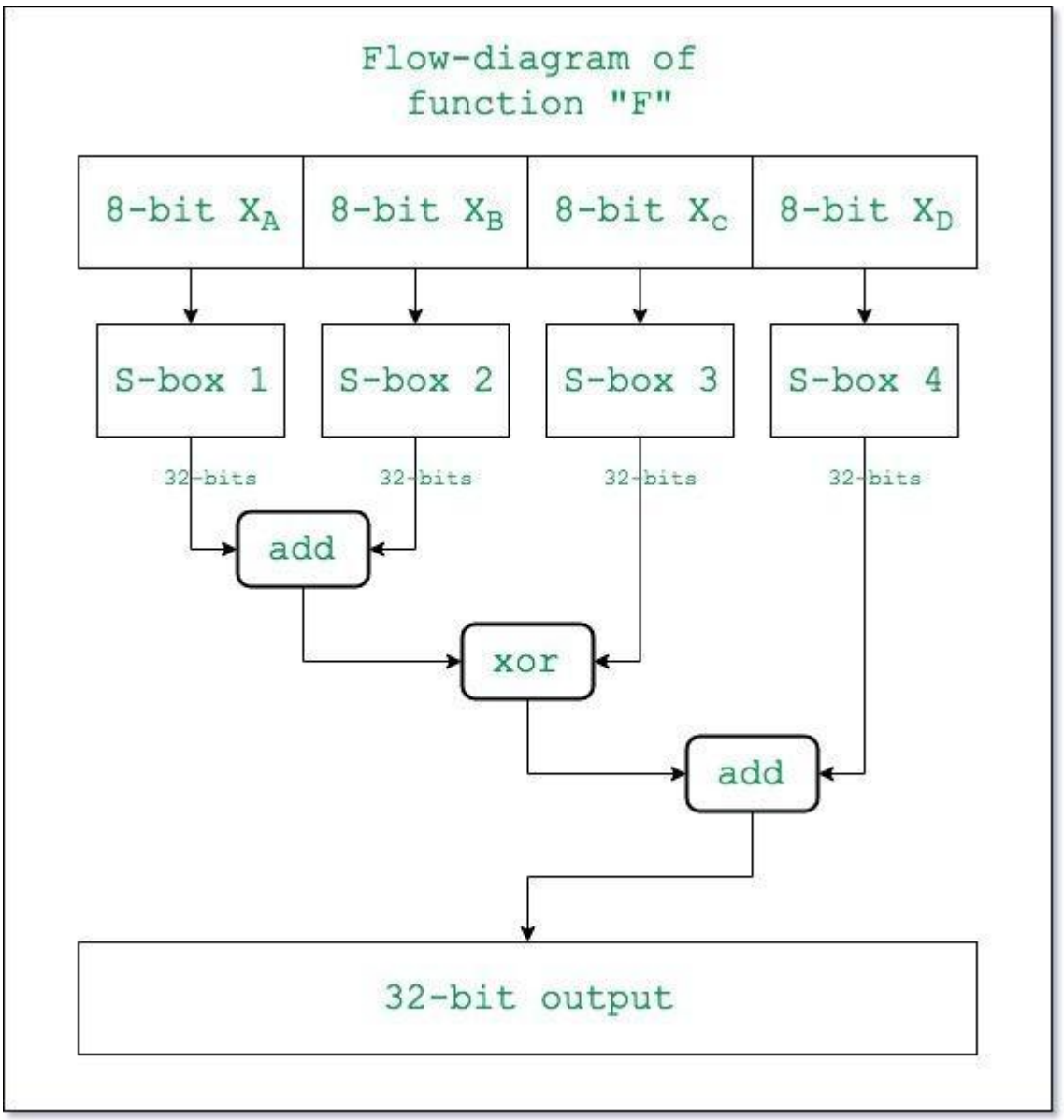


The description of the function ” F ” is as follows:

Here the function “add” is addition modulo 2^32.

b. Post-processing: The output after the 16 rounds is processed as follows:


Flow-diagram of post-
processing step
64-bit output from 16-rounds
32-bit X_L
32-bit X_R
32-bit P[1]
32-bit P[0]
32-bit X_L
32-bit X_R
64-bit cipherText


Flow-diagram of
function "F"
8-bit X_A
8-bit X_B
8-bit X_C
8-bit X_D
S-box 1
S-box 2
S-box 3
S-box 4
32-bits
32-bits
32-bits
32-bits
add
xor
add
32-bit output

	DES	3DES	Blowfish	AES
Key length	56 bits	112 or 168 bits	448 bits	128, 192, or 256 bits
Block Size	64 bits	64 bits	64 bits	128bits
Developped in	1975	1978	1993	2000
Speed	Slow	Slow	Fast	Fast
Security	Not secure enough	Not secure enough	Secure enough	Excellent security
Structure	Feistel	Feistel	Feistel	Substitution Permutation
Time Required to Check All Possible Keys at 50 billion Keys per second	400 days	800 days	~3200 days	5 x $10^{21} days$

RSA Encryption Algorithm

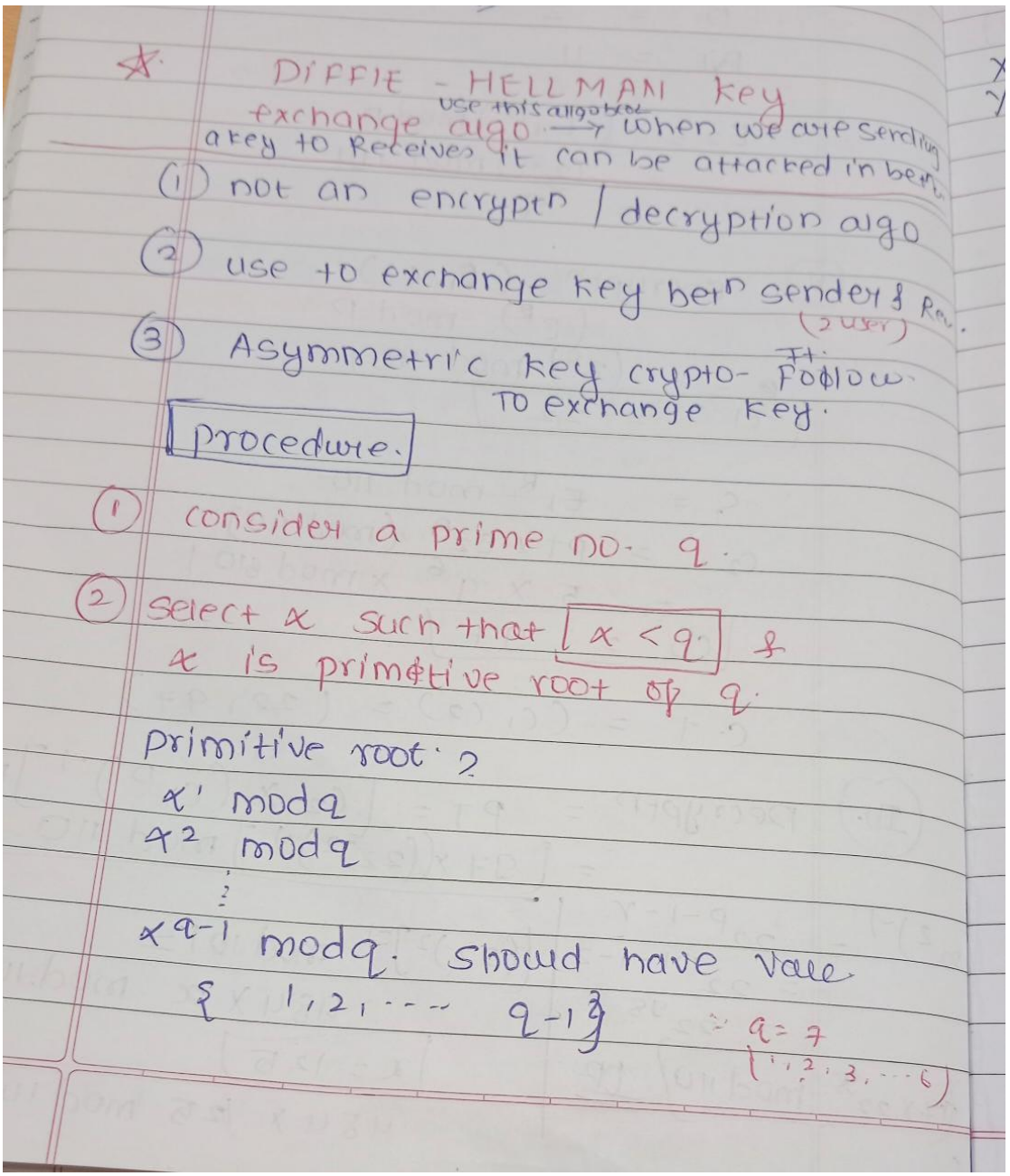
RSA encryption algorithm is a type of public-key encryption algorithm. To better understand RSA, lets first understand what is public-key encryption algorithm.

Public key encryption algorithm:

Public Key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which sender and receiver use different keys for encryption and decryption. Each sender is assigned a pair of keys:

- **Public key**
- **Private key**

The **Public key** is used for encryption, and the **Private Key** is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using user's public key. But only the user can decrypt the message using his private key.



DIFFIE - HELLMAN key exchange algo

use this algorithm → when we are sending a key to receiver it can be attacked in between.

1. not an encrypt / decryption algo
2. use to exchange key betn sender & Rec. (2 user)
3. Asymmetric key crypto- It follow. To exchange key.

Procedure.

1. consider a prime no. q.
2. select α such that $\alpha < q$ & α is primitive root of q.

primitive root ?

$\alpha^1 \bmod q$

$\alpha^2 \bmod q$

⋮

$\alpha^{q-1} \bmod q$. should have value $\{1, 2, \cdots q-1\}$

∴ $q = 7$ $(1, 2, 3, \cdots 6)$

(3) X_A (given) (private value of A) and $X_A < q$

calculate $Y_A = \alpha^{X_A} \bmod q$

X = private key
Y = public k.

with help of prvt ky find public k. (key)

(4) X_B (given) (Prvt key OF B) & $X_B < q$

calculate $Y_B = \alpha^{X_B} \bmod q$.

(5) calculate secret keys k_1 & k_2.

For exchanging.

k_1 = person A
k_2 = person B.

$$k_1 = (y_B)^{X_A} \bmod q$$

$$k_2 = (y_A)^{X_B} \bmod q$$

After calculating if $k_1 = k_2$ then success.

if $k_1 = k_2$ than person A & B can exchange the values.

Hence key exchanged successfully