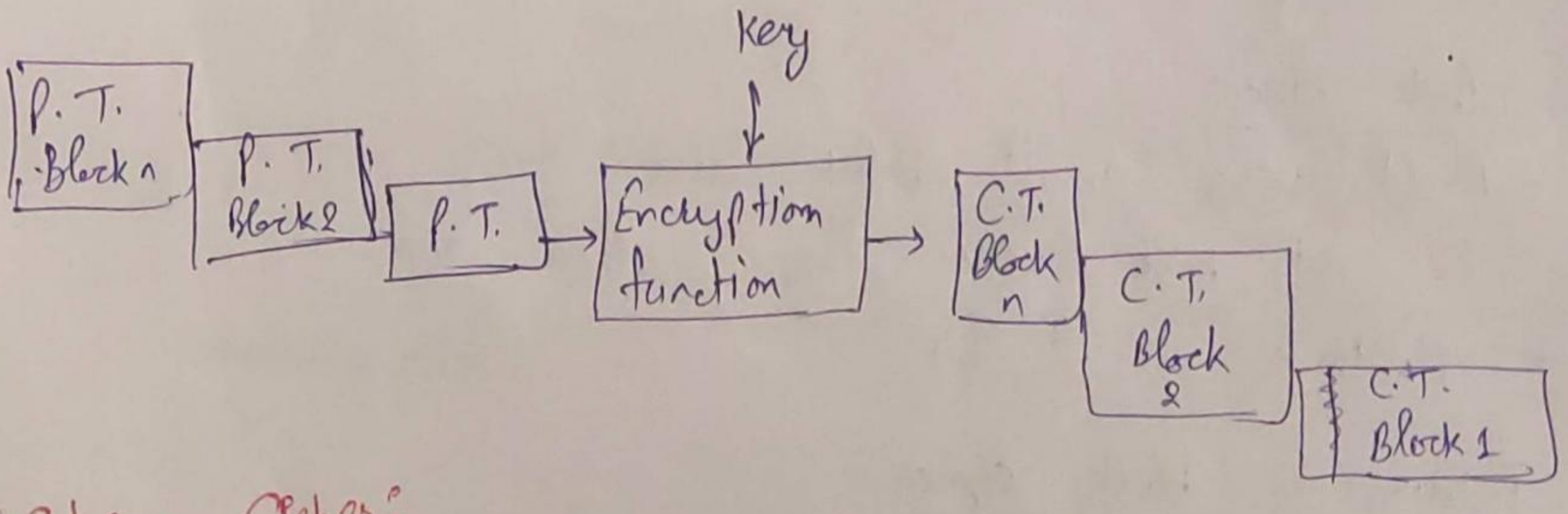
UNIT - II

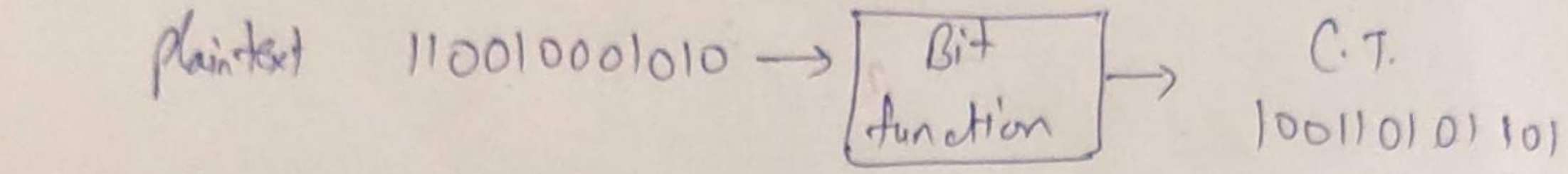
Modern Block Ciphers

Digital data is represented in strings of binary digits (bits) unlike alphabet. Modern cryptosystems need to process this binary strings to convert in to another binary string.

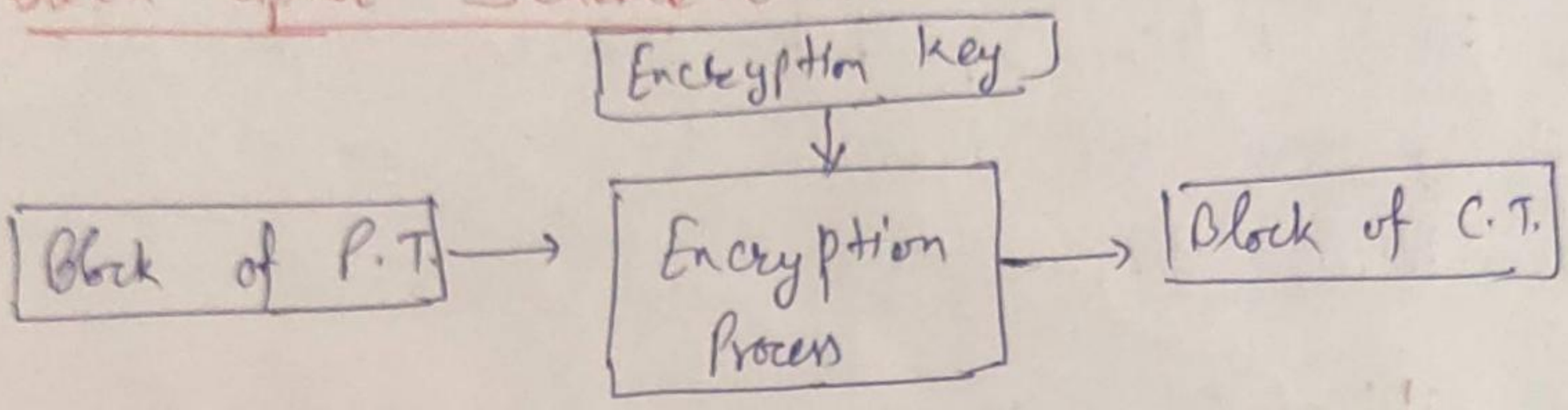
* **Block Ciphers:** The plain binary text is processed in blocks of bits at a time i.e. a block of plaintext bits is selected, a series of operations is performed on this block to generate a block of ciphertext bits. The no. of bits in a block is fixed. eg. DES and AES have block size of 64 and 128, respectively.



* **Stream Cipher:** the plaintext processed one bit at a time i.e. one bit of plaintext is taken, and a series of operations is performed on it to generate one bit of ciphertext. Stream ciphers are block ciphers with a block size of one bit.



* Block Cipher Scheme :- (Basic)



A block cipher takes a block of p.T. bits and generates a block of C.T. bits., generally of same size. The size of block is fixed in the given scheme. The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

Block size :-

- Avoid very small block size.
- Do not have very large block size.
- Multiples of 8 bit.

Padding in block cipher :-

150 bit P.T. provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits need to be padded up with redundant info. So that length of final block equal to block size of the scheme. Now 22 bits need have additional 42 redundant bits added to provide a complete block. This process is called as padding.

Block cipher Schemes:

↳ **Digital Encryption Standard (DES):** Due to its small key size, it is now considered as a 'broken block cipher'.

↳ **Triple DES:** It is a variant scheme but inefficient compared to the new faster block ciphers.

↳ **Advanced Encryption Standard (AES):** It is a relatively new block cipher based on the encryption algo.

↳ **IDEA:** It is a strong block cipher with a block size of 64 and a key size of 128 bits.

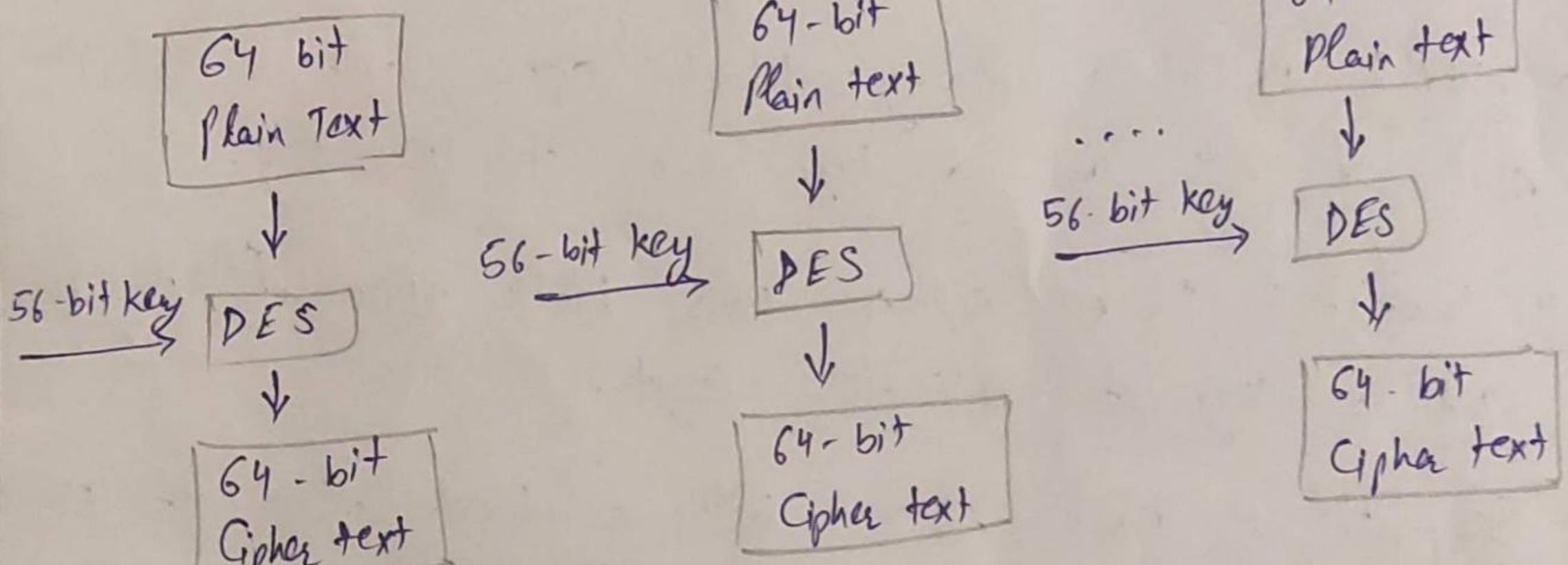
↳ **Twofish:** block cipher uses block size of 128 bits and a key of variable length.

↳ **Serpent:** A block cipher with a block size of 128 bits and key length of 128, 192 or 256 bits. It is slower but more secure design than other block cipher.

* Data Encryption Standard (DES):

DES is a symmetric key block cipher, also known as Data Encryption Algorithm (DEA) by ANSI and DEA-1 by ISO.

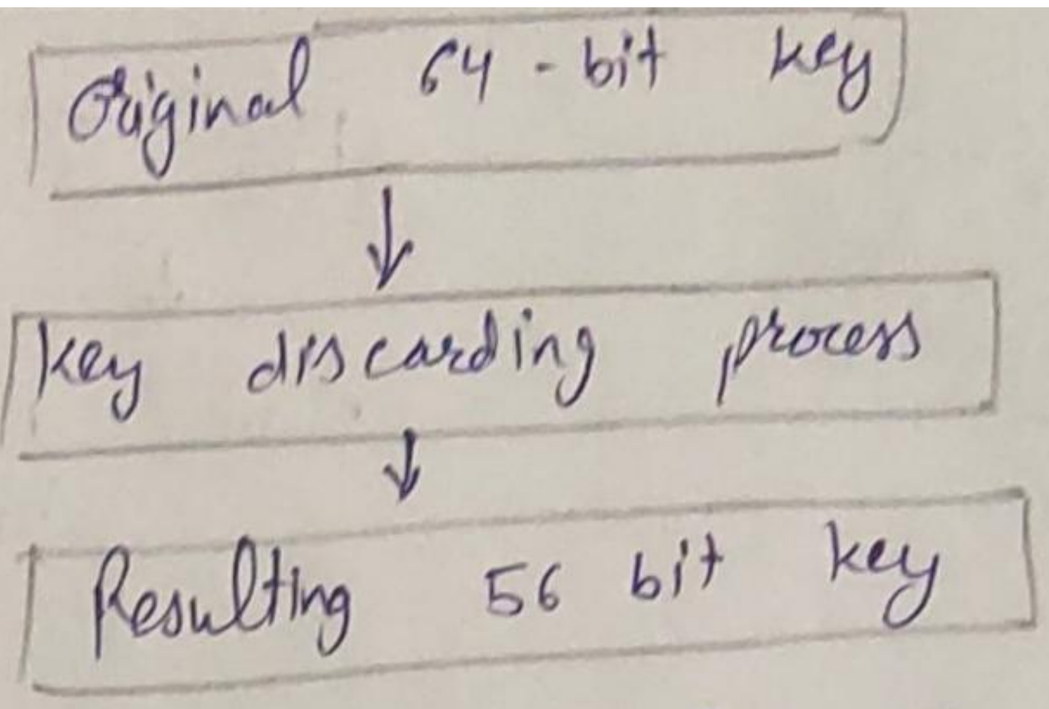
Basic Principle: DES is a block cipher. It encrypts data in blocks of size 64 bit each. That is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The key length is 56 bits.



[Working of DES]

Actually, initial key consists of 64 bits. Before DES process start, every eight bit of the key like 8 bit position 8, 16, 24, 32, 40, 48, 56 and 64 are discarded to produce a 56 bit key.

Thus, discarding of every 8th bit of key produces a 56 bit key from the original 64 bit key.



[key discarding Process]

DES is based on two fundamental attributes of cryptography: substitution / Confusion and transposition/ diffusion. DES consist of 6 steps, each of which is called as round. Each round performs the steps of substitution and transposition.

Step1) 64-bit plain text block is handed over to an Initial Permutation (IP) fun".

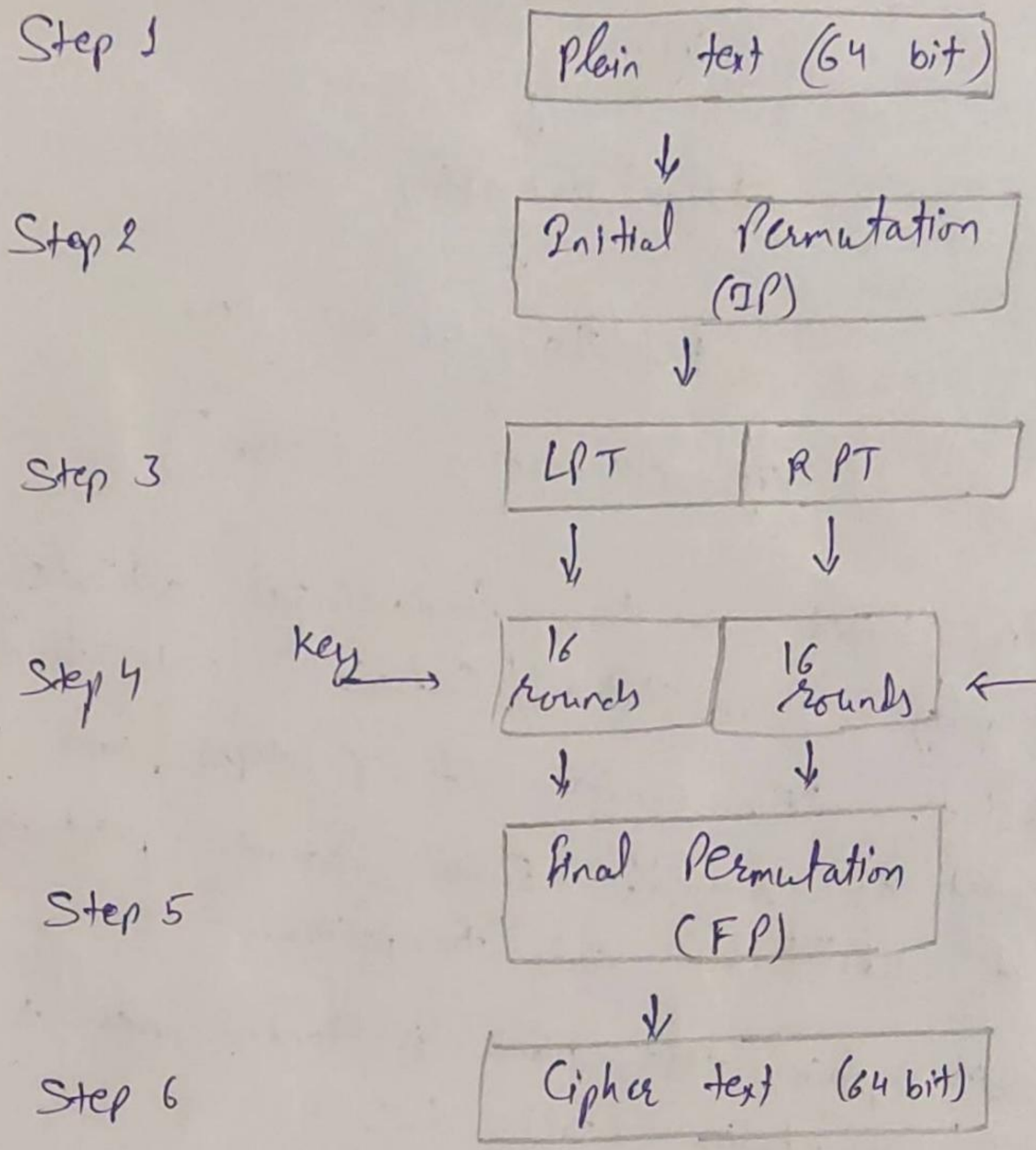
Step2) The IP is performed on P.T.

Step3) IP produces two halves of permuted block; Left Plain text (LPT) and Right Plain text (RPT)

Step4) Now each of LPT and RPT go through 16 rounds of encryption process.

Step5) In the end, LPT and RPT are rejoined and a Final permutation (FP) is performed on the combined block.

Step6) Result of this process produces 64-bit cipher text.



Initial Permutation (IP): IP happens once and before the first round.

Eg: IP replaces the first bit of original plain text block with 58th bit. second bit of original plain text block with 50th bit. This is nothing but jugglery of bit position of original plain text block.

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

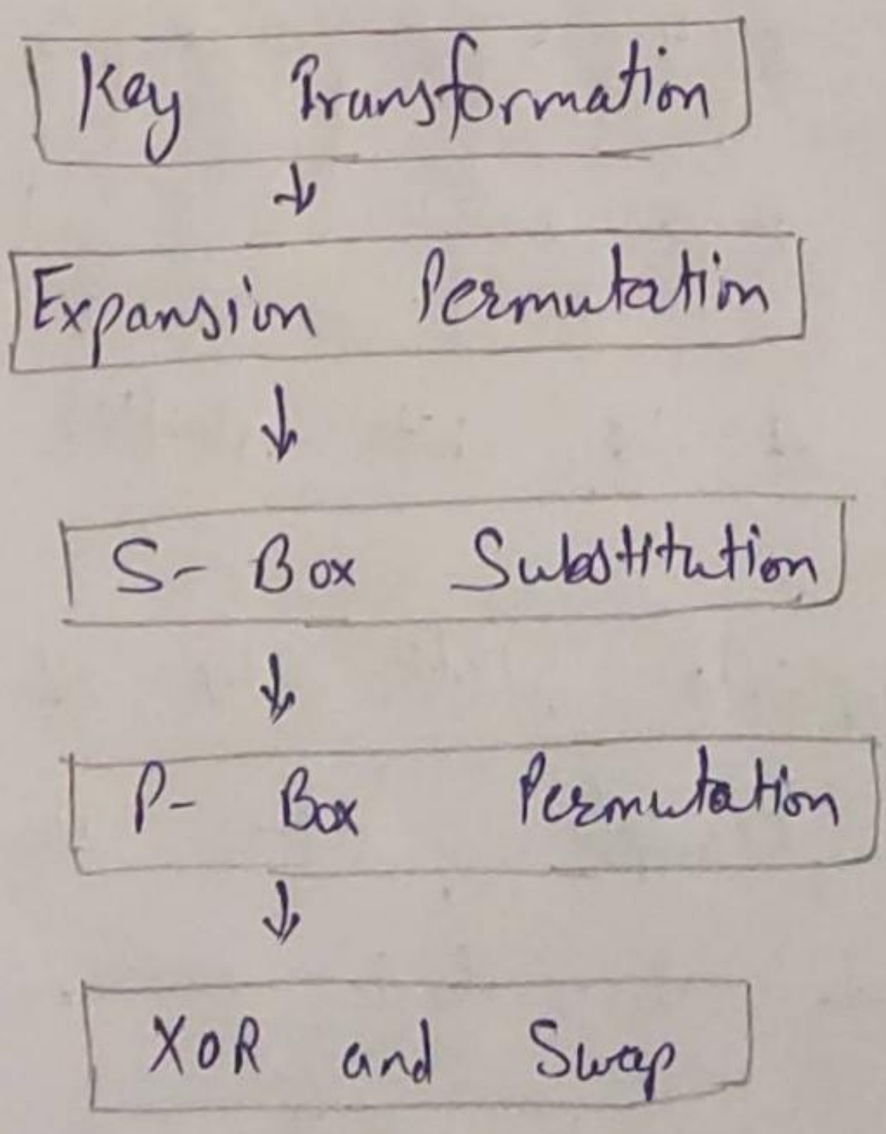
[original plain text]

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

[initial Permutation (IP)]

After IP, resulting 64-bit permuted text block is divided into two half blocks. Each block consist of 32 bits.. (LPT & RPT blocks). Now 16 rounds are performed on these two blocks.

Rounds:



[Details of One round in DES]

Step 1: Key transformation: the initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of initial key. Thus, for each round, a 56-bit key is available.

From this 56-bit key, a different 48-bit sub-key is generated during each round using a process called as key transformation.

For this, 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

Eg: if the round no. is 1, 2, 9 or 16, the shift is done by only 1 position. For other rounds, the circular shift is done by two positions.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
No. of key bit shifted.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

[No. of key bits shifted per round]

After an appropriate shift, 48 of the 56 bits are selected. for example, after the shift, bit no. 14 moves into the first position, bit no. 17 moves into the second position and so on.

This table contains only 48 bits positions. Bit no. 18 is discarded (not find in the table), like 7 others. Total 8 bits are discarded. Since, the key transformation process involves permutation as well as selection of a 48 bit sub-set of the original 56 bit key, it is called as Compression permutation.

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

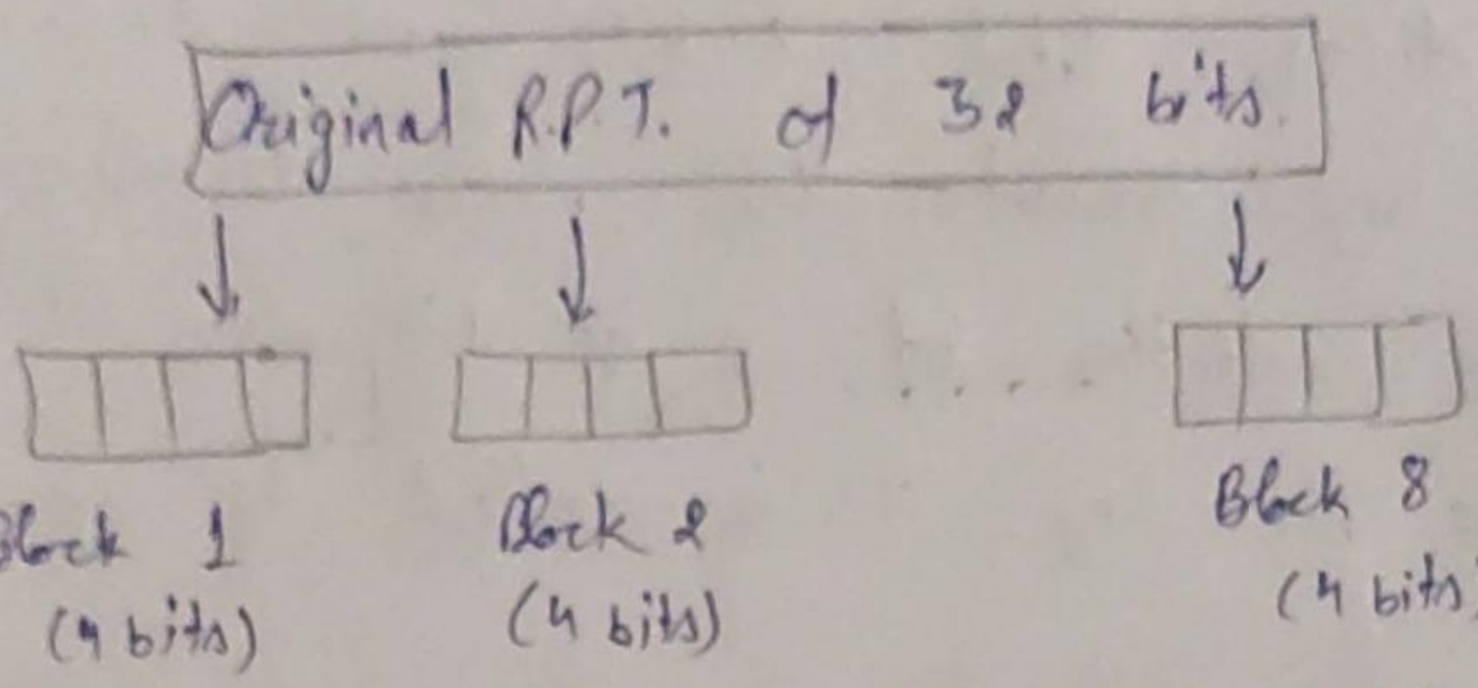
[Compression Permutation]

Because of this compression permutation technique, a diff. subset of key bits is used in each round. That makes DES not so easy to crack.

Step 2: Expansion Permutation: After initial permutation, we had two 32-bits P.T. areas, called as LPT and RPT.

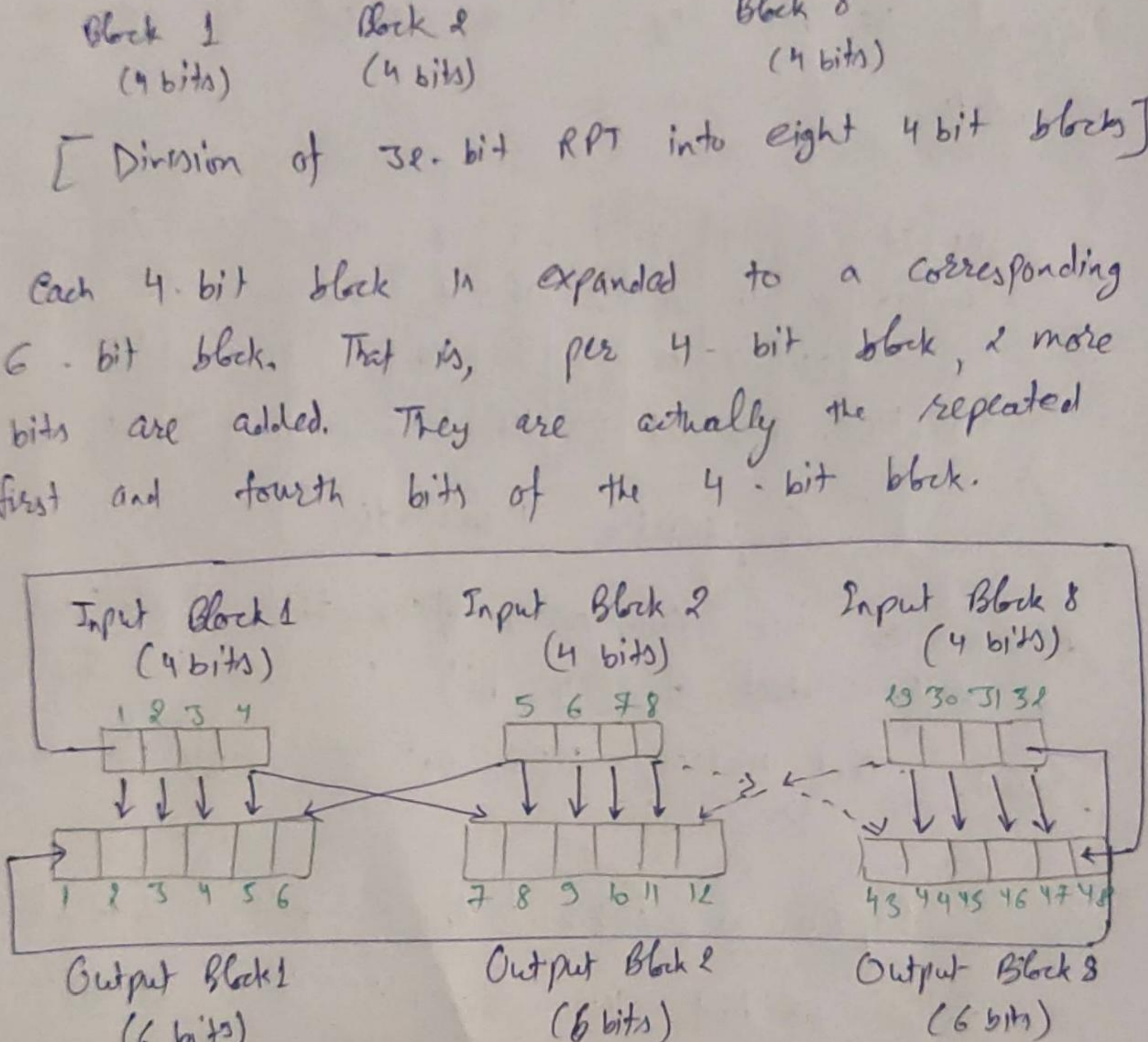
During expansion permutation, the RPT is expanded from 32 bits to 48 bits. The process as follows:

1) The 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits.



[Division of 32-bit RPT into eight 4 bit blocks]

2) Each 4-bit block is expanded to a corresponding 6-bit block. That is, per 4-bit block, 2 more bits are added. They are actually the repeated first and fourth bits of the 4-bit block.



[RPT expansion permutation process]

As we can see, the first input bit goes into the second and 48th output position. The second input bit goes into the third output position & so on.

32	1	2	3	4	5	4	5	6	7	8	9
8	9	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	30	31	32	1

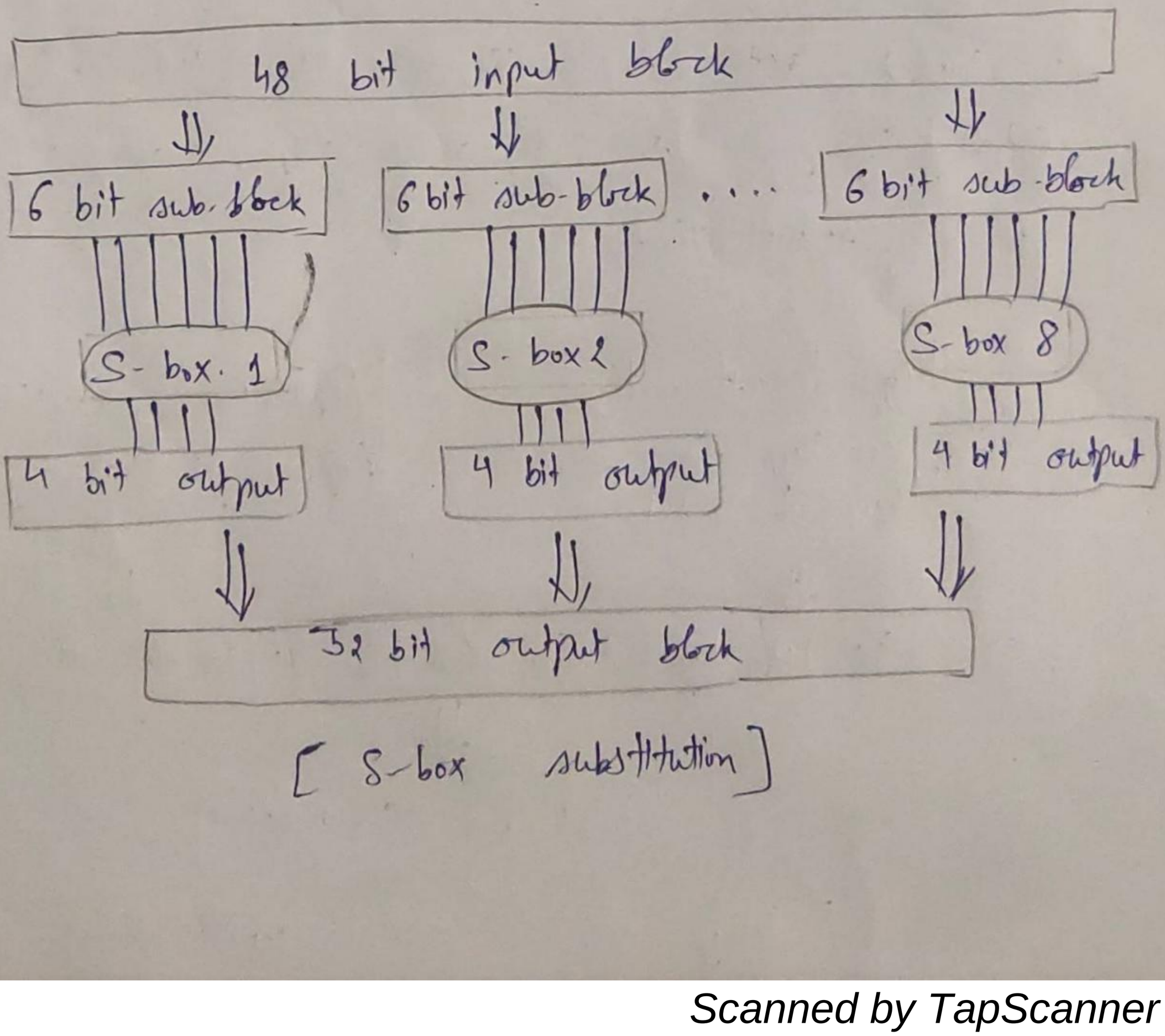
[RPT expansion permutation Table]

Now As we have seen, firstly, the key transformation process compresses the 56-bit key to 48 bits. Then the Expansion permutation process expands the 32-bit RPT to 48 bits. Now, the 48-bit key is XoRed with the 48-bit RPT and resulting output is given to the next step, which is the S-box substitution. . . . . .

[Key to S-box Substitution]

Step 3: S-box substitution: It is a process that accepts the 48 bit input from the XOR operation involving the Compressed key and expanded RPT and produces 32 bit output using the substitution technique. The substitution is performed by eight substitution boxes. (S-boxes).

Each of the eight S-boxes has a 6-bit input and a 4-bit output. The 48 bit input block is divided into 8 sub-blocks (each containing 6 bits) and each such sub-block is given to an S-box. The S-box transform the 6 bit input into a 4 bit output.



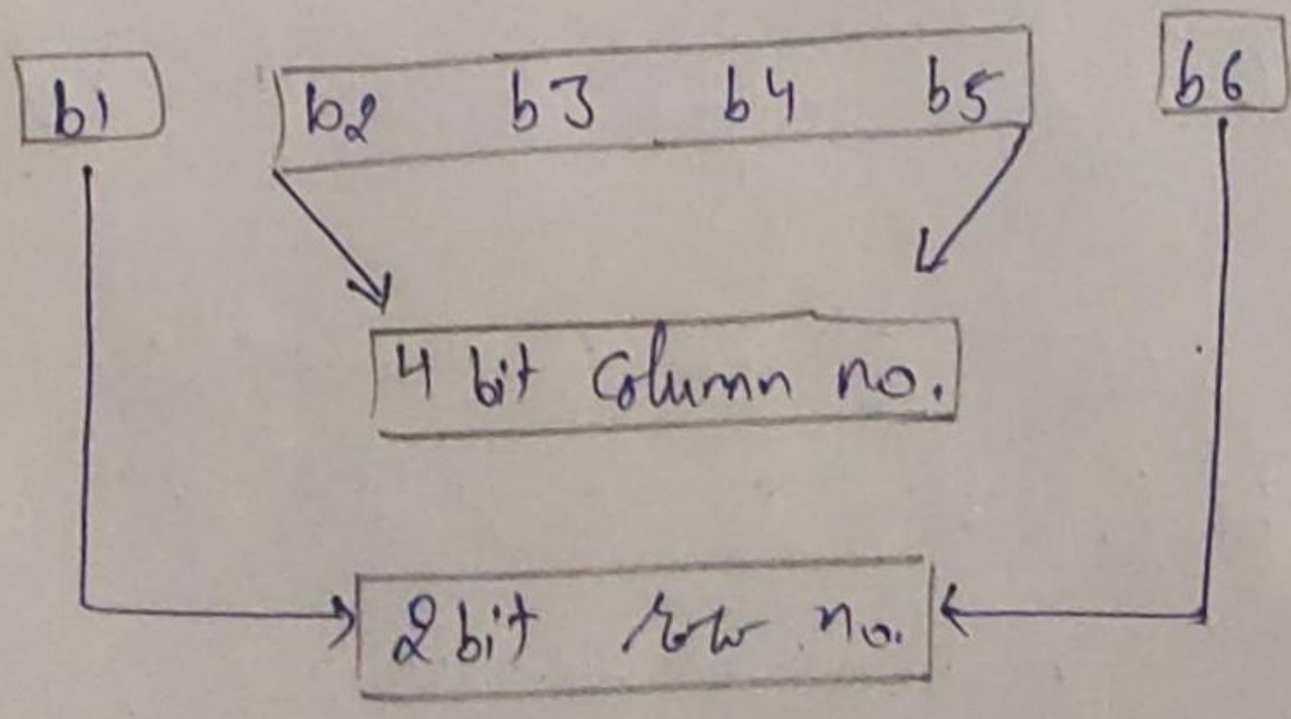
[S-box substitution]

S-box has 4 rows (0 to 3) and 16 columns (0 to 15).
Thus we have 8 tables, one for each S-box.
(Page No 106 fig 3.3 of Atul Kahate)

6 bit input indicates which row and column and therefore, which intersection is to be selected thus, determining the 4-bit output.

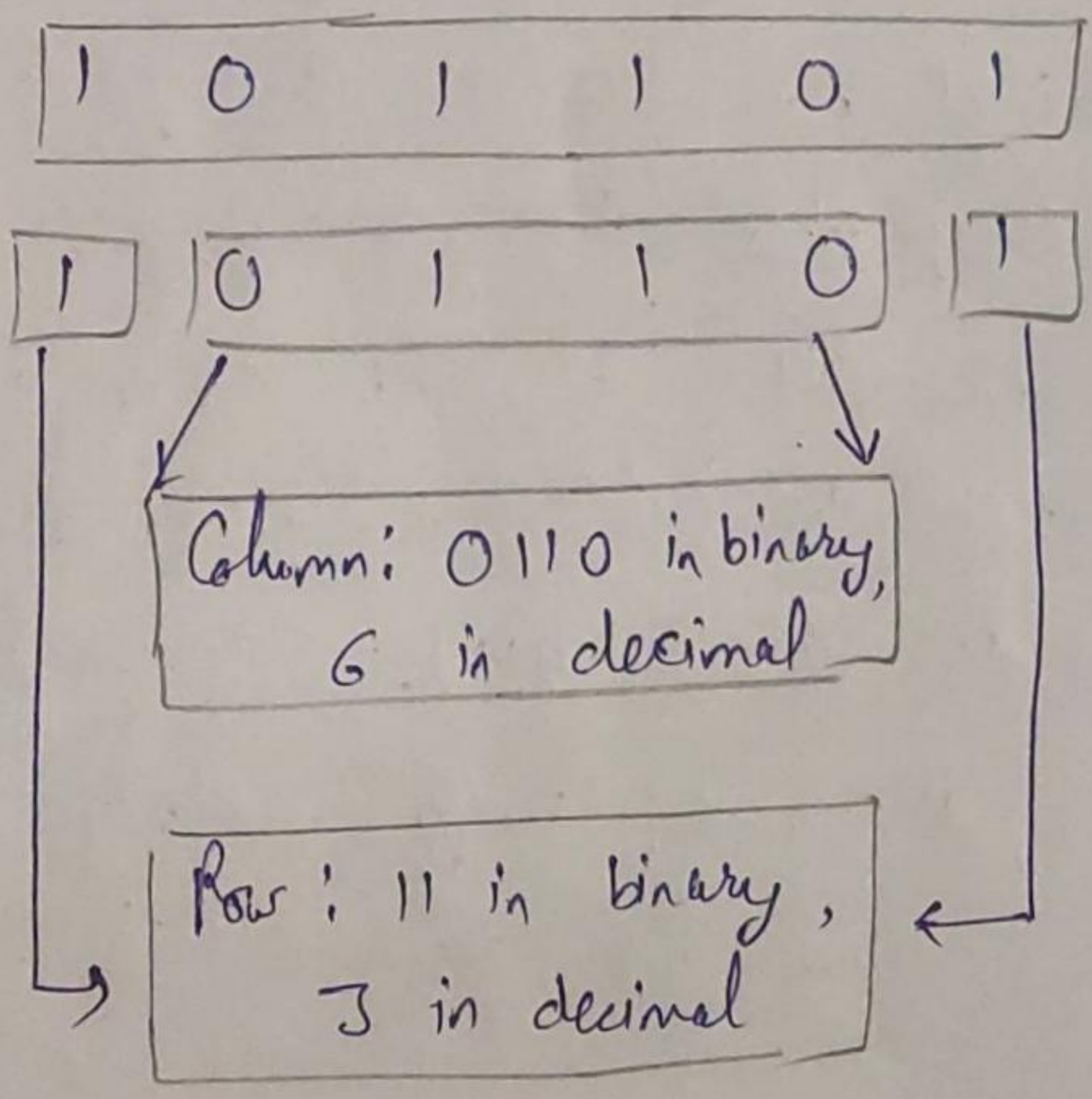
Eg: Let assume that six bits of S-box are indicated by b_1, b_2, b_3, b_4, b_5 and b_6. Now bits b_1 and b_6 are combined to form a two-bit no. Two bits can store any decimal no. b/w 0 (binary 00) and 3 (binary 11). This specifies the row no. The remaining four bits b_2, b_3, b_4, b_5 make up a four-bit no., which specifies the Col^n no. b/w decimal 0 (binary 0000) and 15 (binary 1111).

Thus, 6-bits input automatically selects the row no. and column no. for the selection of the O/P.



[Selecting an entry in a S-box based on 6-bit input]

Eg. Suppose the bit 5 to 8 of the 48 bit input (i.e. input to the second S-box) contain a value 101101 in binary. Therefore, we have $(b_1, b_6) = 11$ and $(b_2, b_3, b_4, b_5) = 0110$. Thus, the o/p of S-box 2 at the intersection of row 3 $[b_1, b_6] = 11$ and Column no. 6 $[b_2, b_3, b_4, b_5] = 0110$ will be selected, which is 4. [Count row & Column from 0, not 1)



[Selection of S-box output based on input]

The o/p of each S-box is then combined to form a 32-bit block, which is given to the last stage of round, the P-box permutation.

Step 4: P-box permutation: The o/p of S-box consist of 32 bits. These 32 bits are permuted using a P-box. This is simple permutation (i.e. replacement of each bit with another bit, as specified in the P-box table, without any expansion or compression). This is called as P-box permutation.

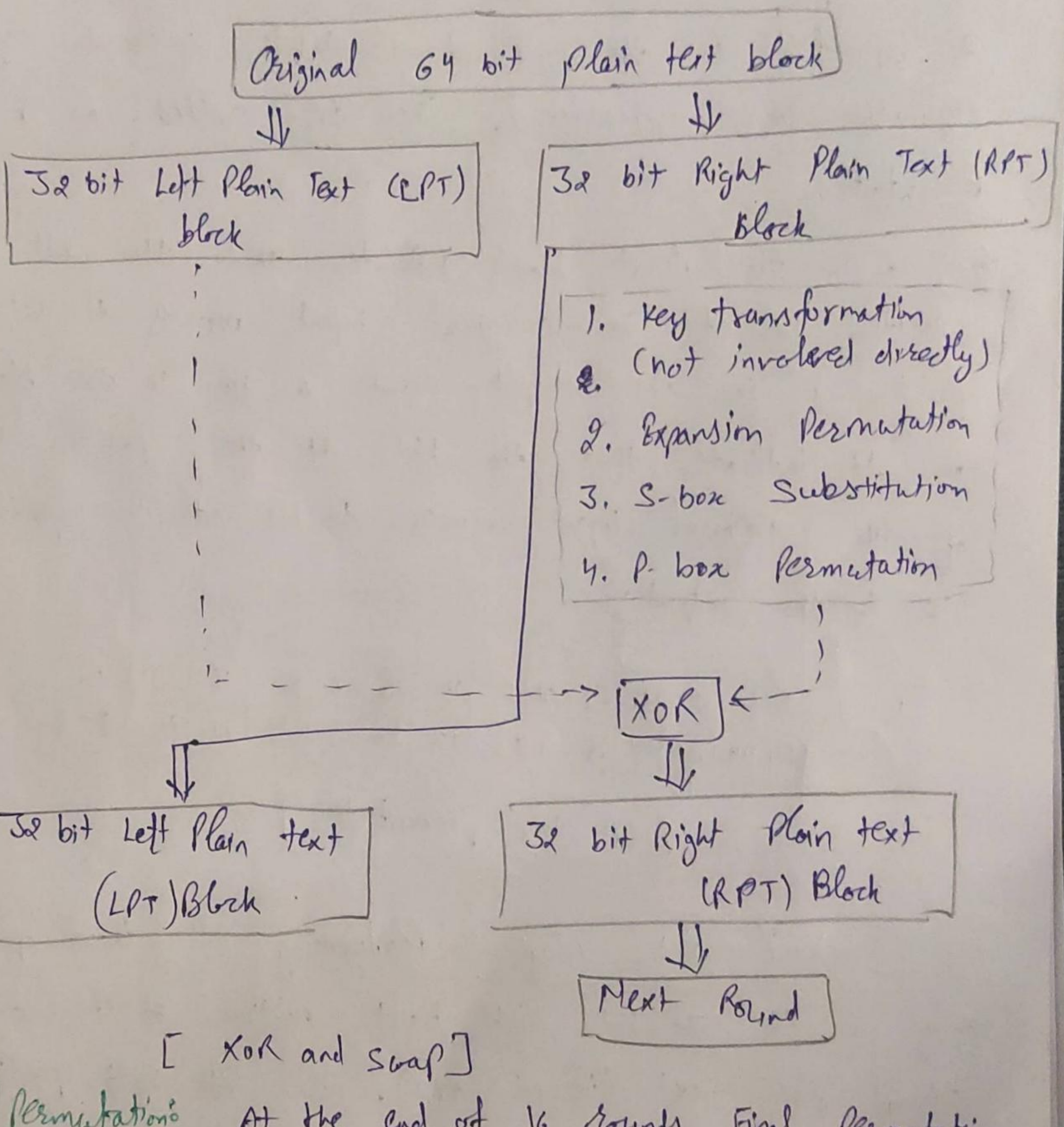
Eg. 16 in the first block indicates that the bit at position 16 of the original input moves to bit at position 1 in the output and a 10 in the block no. 16 indicates that the bit at the position 10 of the original input moves to bit at the position 16 in the output.

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

[P-box permutation]

Step 5: XoR and Sway: Performing all these operations only on the 32-bit right half portion of the 64-bit original plain text (i.e on RPT). The left half portion (i.e LPT) was untouched. The left half portion of the initial 64 bit plain text block (ie. LPT) is XoRed with the output produced by P-box permutation.

The result of this XOR operation becomes the new right half (i.e. RPT). The old right half (RPT) becomes the new left half, in a process of swapping.



[XOR and swap]

Final Permutation: At the end of 16 rounds, Final Permutation is performed (only once). for instance, 40th input bit takes the position of the 1st output bit & so on. (Page No. 110) (Fig. 3.37)

The output of the final permutation is 64 bit encrypted block.

* Strength of DES:

With a key length of 56 bits, there are 2^{56} possible keys, which is appox. 7.2×10^{16} keys. Thus, a brute-force attack appears impractical.

* Block Cipher Principles:

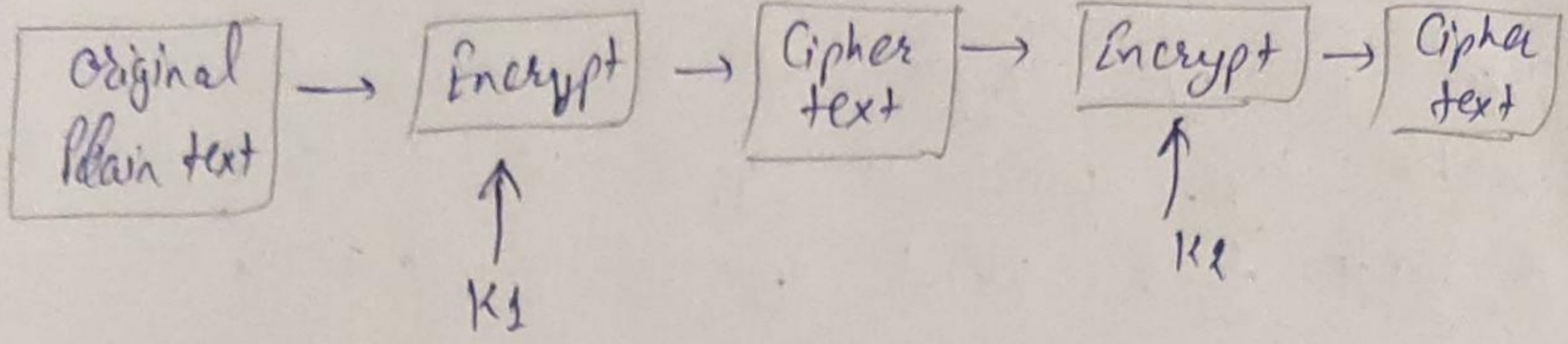
1) **Number of Rounds:** The no. of rounds judges the strength of block cipher algo. If no. of rounds are more than difficult is for cryptanalysis to break the algo.

2) **Design of Function F:** The fun^n F of the block cipher must be designed such that it must be impossible for any cryptanalysis to unscramble the substitution. (to restore to intelligible form).
If the fun^n F is more non-linear, then it would be difficult to crack it.

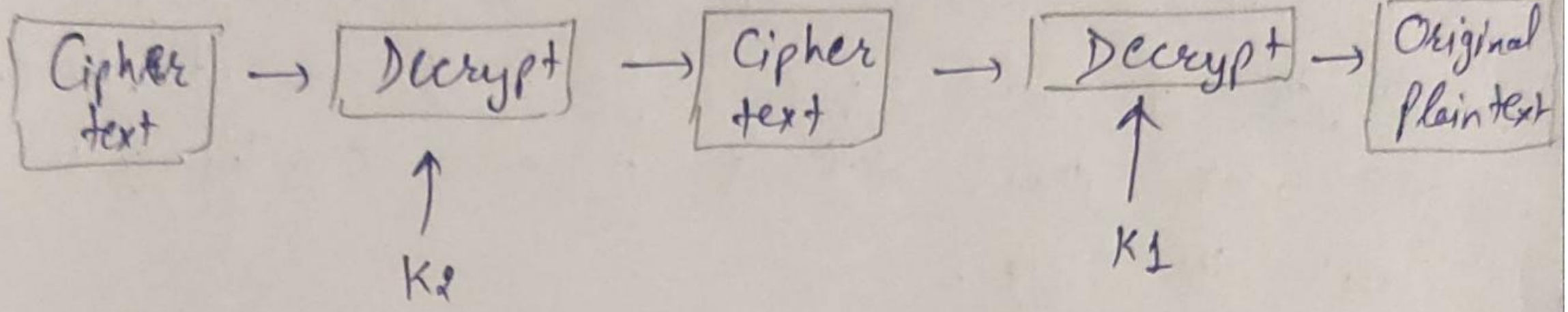
3) **Key Schedule Algo:** key schedule confirm the strict avalanche effect and bit independence criterion.

* Variations of DES:

1) **Double DES:** Double DES uses two keys, say K1 and K2. It first perform DES on the original plain text using K1 to get the encrypted text. It again performs DES on the encrypted text but this time with the other key, i.e. K2.



[Double DES encryption]

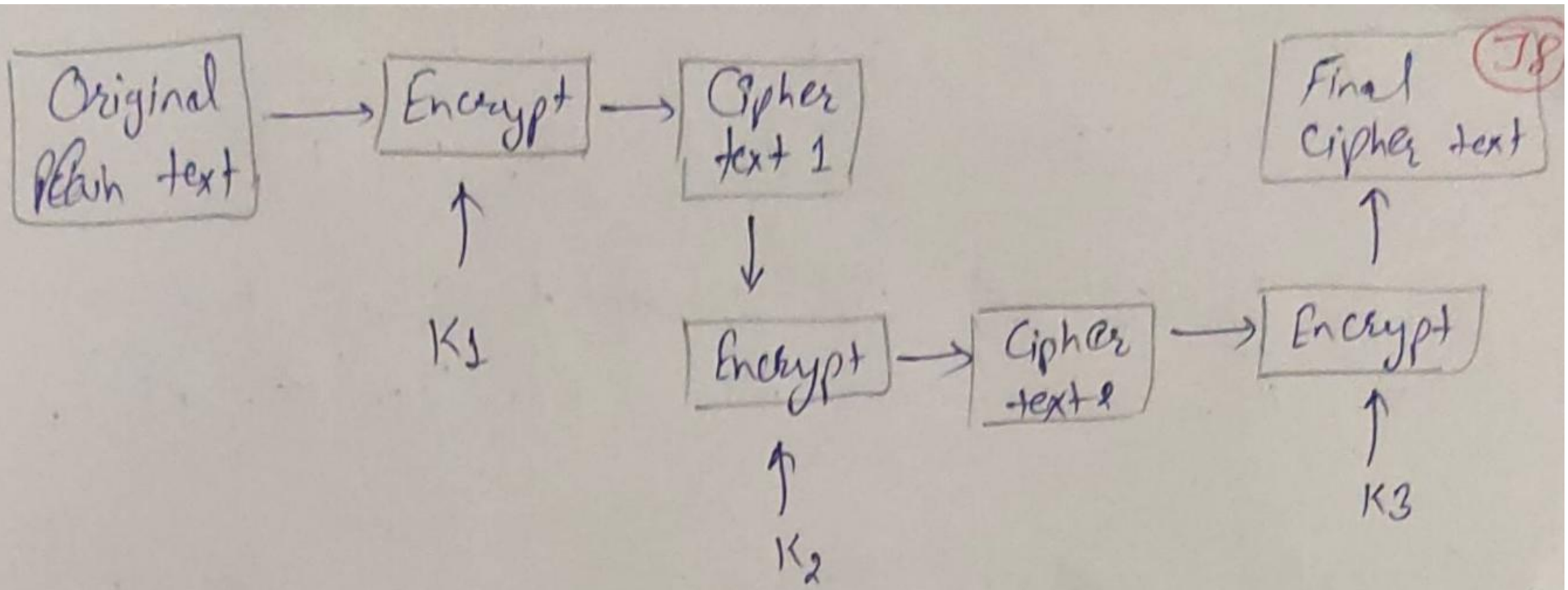


[Double DES decryption]

The doubly encrypted cipher text block is first decrypted using the key K2 to produce the singly encrypted cipher text.

This cipher text block is then decrypted using the key K1 to obtain the original plain text block.

2) Triple DES with three keys: the plain text block P is first encrypted with a key K1, then encrypted with a second key K2 and finally with a third key, K3, where K1, K2 and K3 are all diff. from each other.



To decrypt the cipher text C and obtain the plain text P, we need to perform the operation

$$P = D_{K3}(D_{K2}(D_{K1}(C))).$$

Triple DES with Two keys:- Triple DES with three keys is highly secure. It can be ~~encoded~~ denoted in the form of equation as

$$C = E_{K3}(E_{K2}(E_{K1}(P)))$$

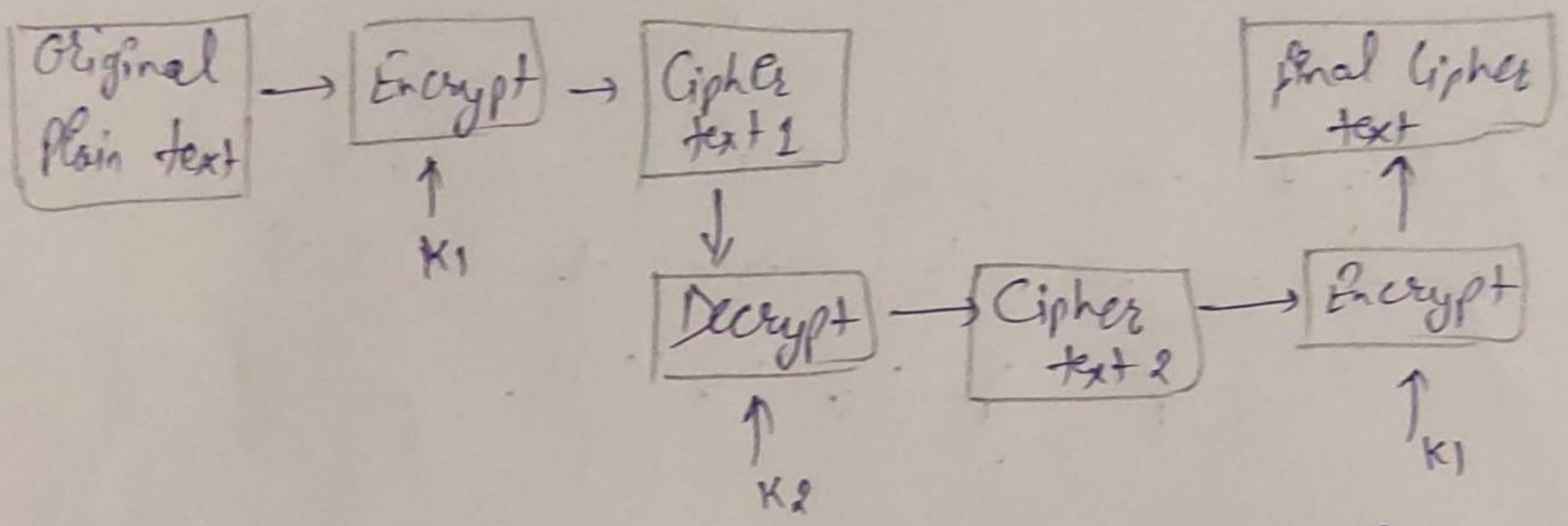
However, Triple DES with three keys also has the drawback of requiring $56 \times 3 = 168$ bits for the key.

Tuchman uses two keys for triple DES. Algorithm as follows:-

1) Encrypt the plain text with key K1. Thus, we have $E_{K1}(P)$.

2) Decrypt the output of Step 1 above with key K_2. Thus we have $D_{K_2}(E_{K_1}(P))$.

3) Finally, Encrypt the output of Step 2 again with key K_1. Thus, we have $E_{K_1}(D_{K_2}(E_{K_1}(P)))$.



To decrypt cipher text C and obtain plain text P, we need to perform the operation

$$P = D_{K_1}(E_{K_2}(D_{K_1}(C))).$$

Triple DES is also called as Encrypt - Decrypt - Encrypt (EDE) mode.

* Advanced Encryption Standard (AES):

1) Introduction: AES is introduced because of weakness of DES. The 56 bits keys of DES were no longer considered safe against attacks based on exhaustive key searches and 64 bit blocks were also considered as weak. So AES was to be based on 128 bit blocks with 128 bit keys.

~~Cipher text → (64 bits)~~

According to its designers, the main features of AES are as follows:

i) Symmetric & parallel structure: This gives the implementers of the algo. a lot of flexibility. It also stands up well against cryptanalysis attacks.

ii) Adapted to modern processors: The algo. works well with modern processors (Pentium, RISC, parallel).

iii) Suited to Smart Cards: The algo. can work well with smart cards.

Rijndel supports key lengths and plain text block sizes from 128 bits to 256 bits, in steps of 32 bits. AES mandates that the plain text block size must be 128 bits and key size should be 128, 192 or 256 bits.

In general, two version of AES are used: 128 bit plain text block combined with 128 bit key block and 128 bit plain text block with 256 bit key block.

for standard AES, 128 bit plain text block and 128 bit key length are used as a pair. Since 128 bits give a possible key range of 2^{128} or 3×10^{38} keys.

2) Operation: Rijndael also uses the basic techniques of substitution and transposition (i.e permutation.).

→ The key size and plain text block size decide how many rounds need to be executed.

→ The min. no. of rounds is 10 (when key size and plain text block size are each 128 bits) and max. no. of rounds is 14.

→ One key differentiator b/w DES and Rijndael is that all the Rijndael operations involve entire byte and not individual bits of a byte.

Rijndael Steps description:

i) Do the following one-time initialization processes:

a) Expand the 16 byte key to get the actual key block to be used.

b) Do one time initialization of the 16 byte plain text block (called as state).

c) XOR the state with the key block.

ii) For each Round, do the following: (40/50)

a) Apply S-box to each of the plain text bytes.

b) Rotate row k of the plain text block (i.e. state) by k bytes.

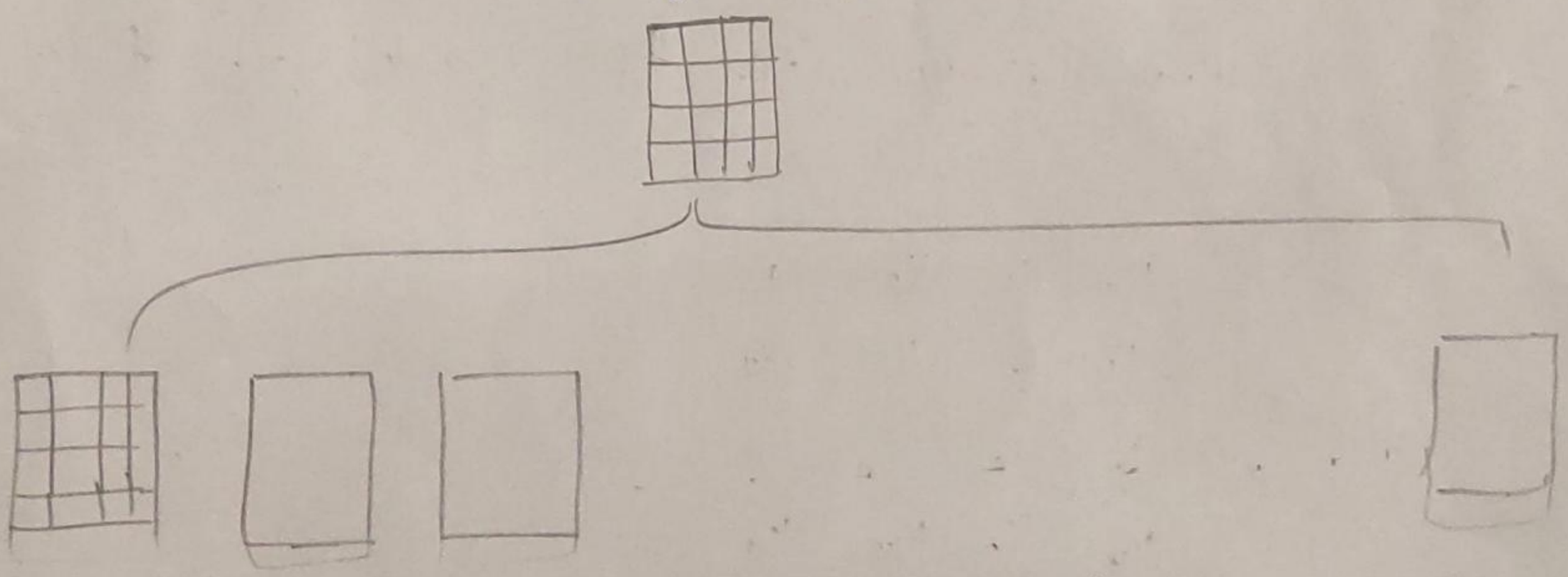
c) perform a mix columns operations.

d) XoR the state with the key block.

3) One - time Initialization Process :

a) Expand the 16-byte key to get the Actual key block to be used? The inputs to the algo. are the key and the plain text. The key size is 16 bytes. This step expands this 16-byte key into 11 arrays., Each array contains 4 rows and 4 columns.

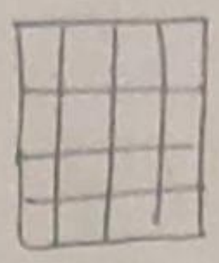
16 byte key



[Expanded into 11 arrays, each of size 4x4]

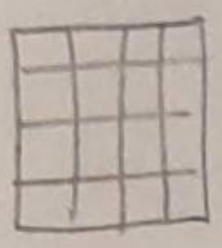
In other words, the original 16-byte key array is expanded into a key containing $11 \times 4 \times 4 = 176$ bytes. One of these 11 array is used in the initialization process and other 10 arrays are used in 10 Rounds. One array per Round.

1) Firstly, original 16 byte key is copied into the first 4 words of the expended key (i.e. the first 4×4 array).



16 byte key

Copied, as it is.

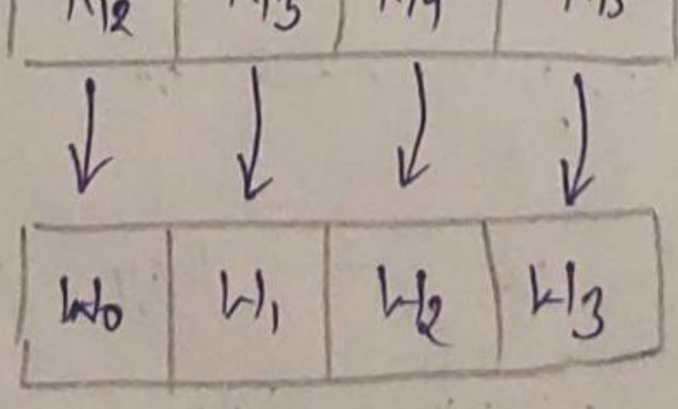


To be expanded into 11 arrays, each of size 4×4

[key expansion : Step 1]

2) After filling the first array (for words numbered W_0 to W_3) of the expanded key block as explained above, the remaining 10 arrays (for words numbered W_4 to W_{43}) are filled one-by-one. Every time one such 4×4 array (i.e. four words) gets filled.

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}



W_0	W_1	W_2	W_3

[key Expansion : Step 1]

Let us understand the three function:
Substitute, Rotate & Constant.

i) Function Rotate performs a circular left shift on the contents of the word by one byte. Thus, if an input word contains four bytes numbered [B1, B2, B3, B4]; then the o/p word would contain [B2, B3, B4, B1]

ii) Function Substitute performs a byte substitution on each byte of the input word. For this purpose, it uses an S-box.

iii) In the fun" Constant, the o/p of the above steps is XORed with a Constant. This Constant is a word, consisting of 4 bytes. The value of the Constant depends on the round no. The last three bytes of a constant word always contain 0.

Thus, XORing any input word with such a constant is as good as XORing only with the first byte of input word. These Constant values per round are listed as:

Round No.	1	2	3	4	5	6	7	8	9	10
Value of Constant to be used in Hex	01	02	04	08	10	20	40	80	1B	36

[values of constant per round, to be used in the Constant function]

b) Do one time Initialization of the 16 byte plain text Block (Called as State): the 16 byte plain text block is copied into a two-dimensional 4x4 array called as State. The order of copying is in the column order.

That is, the first four bytes of the plain text block get copied into the first column of the state array, the next four bytes of the P.T.B. get copied into second column of the state array and so on.

16-byte plain text block

B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16

⇓

B1	B5	B9	B13
B2	B6	B10	B14
B3	B7	B11	B15
B4	B8	B12	B16

[State array]

[Copying of input text block into state array]

c) XOR the State with key block: Now, first 16 bytes (i.e. four words W[0], W[1], W[2] and W[3]) of the expanded key are XORed into the 16 byte state array (B1 to B16 shown above).

Thus, every byte in the state array is replaced by the XOR of itself and the corresponding byte in the expanded key.

At this stage, initialization is complete and ready for rounds.

4) Processes in Each Round:

The following steps are executed 10 times, one per round.

a) Apply S-box to Each of the Plain Text Bytes:

This step is very straightforward. The contents of the state array are looked up into the S-box. Byte by Byte substitution is done to replace the contents of the state array with the respectives entries in the S-box.

Note: only one S-box is used, unlike DES, which has multiple S-boxes.

b) Rotate Row k of the Plain Text Block (i.e. state) by k Bytes:

Here, each of the four rows of the state array are rotated to the left.

- Row 0 is rotated 0 bytes (i.e. not rotated at all.
- Row 1 is rotatey 1 byte.

- Row 2 is rotated 2 bytes.
- Row 3 is rotated 3 bytes.

Original Array			
1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Modified Array			
1	5	9	13
6	10	14	2
11	15	3	7
16	4	8	12

c) Perform a Mix Columns Operations: Now, each column is mixed independent of each other. Matrix multiplication is used. The o/p of this step is the matrix multiplication of the old values and a constant matrix.

d) XOR the State with the Key Block:

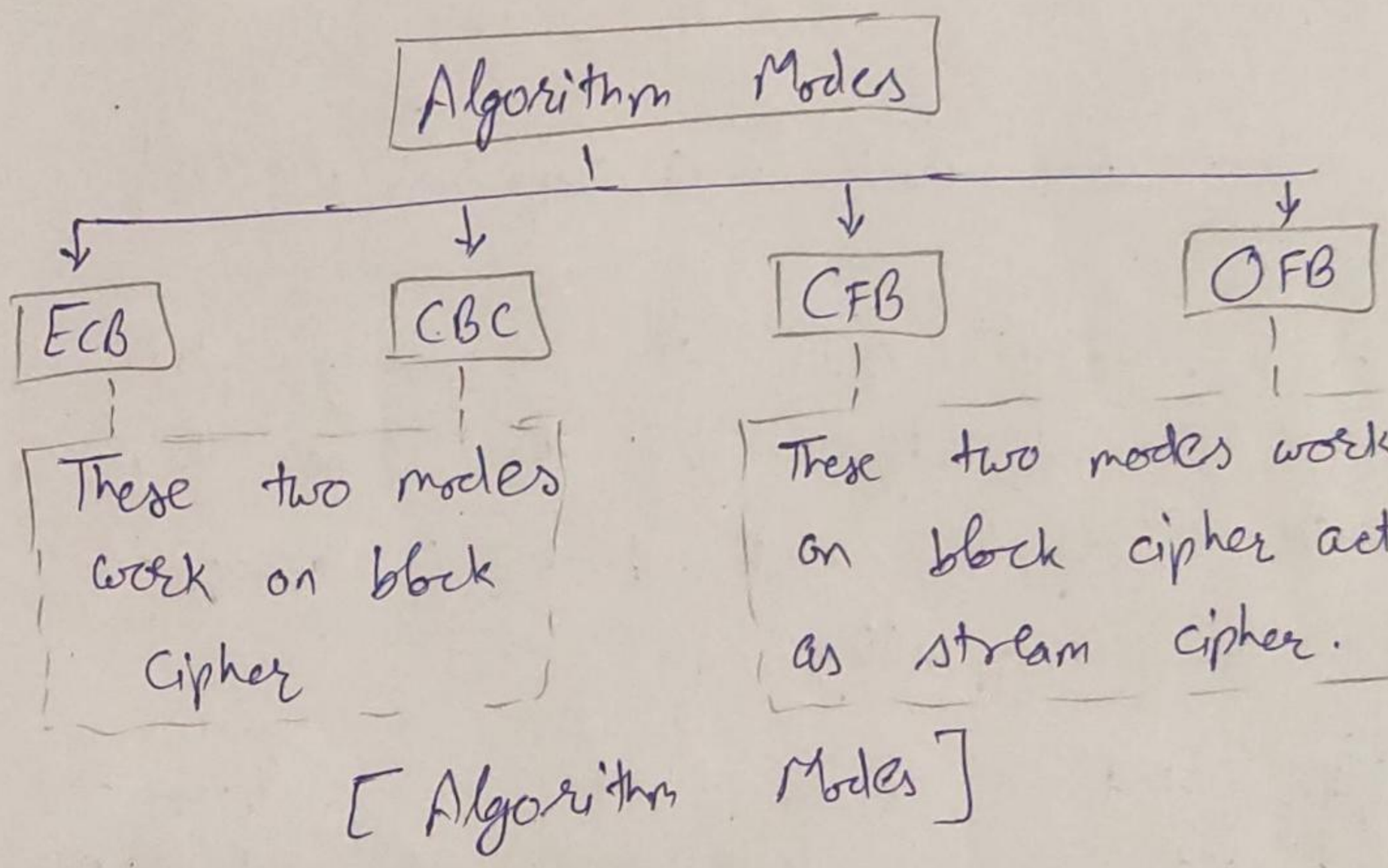
This step XORs the key for this round into the state array.

For decryption, the process can be executed in reverse order.

* Algorithm Modes :

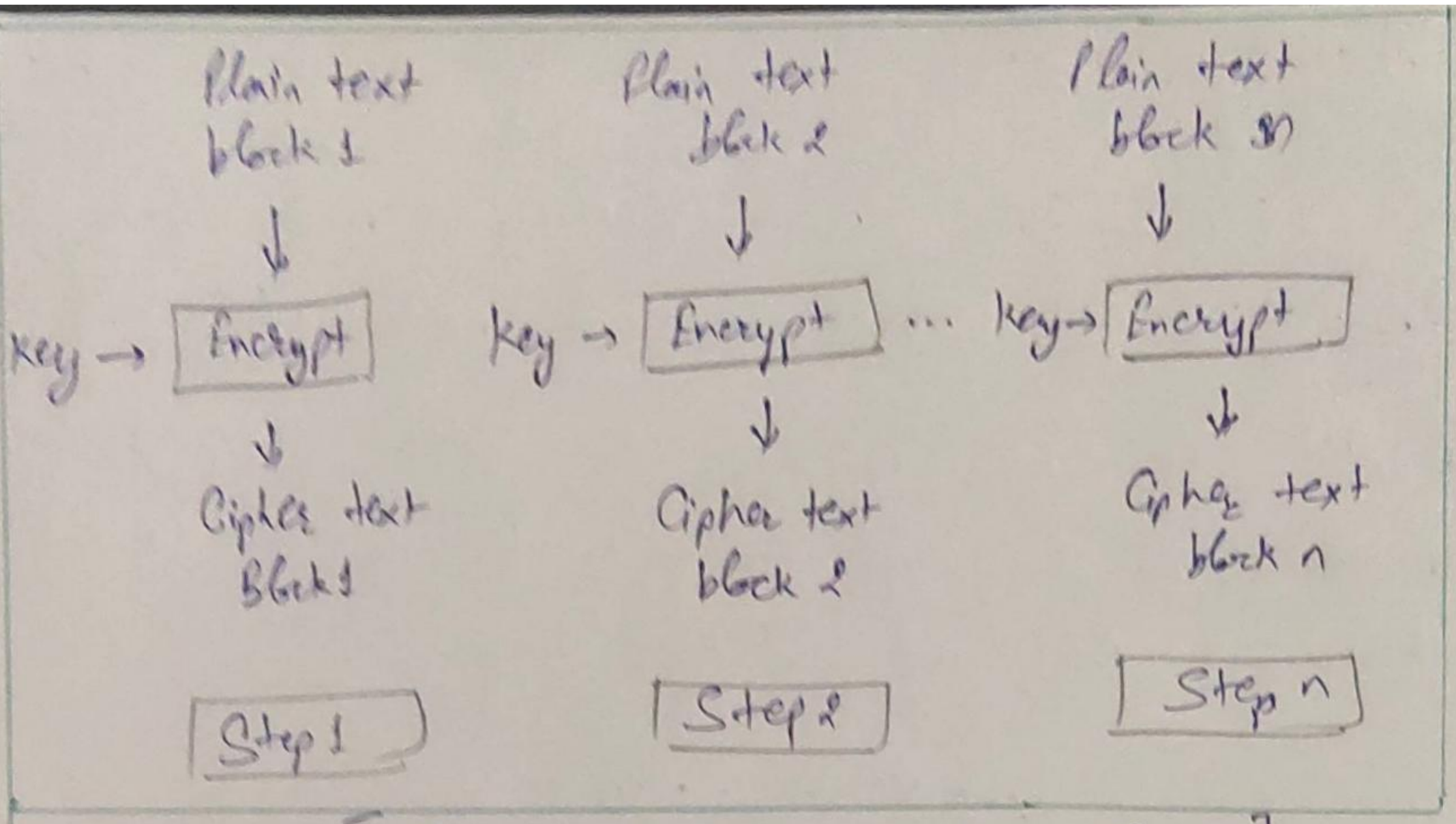
four imp. algorithm modes are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and output Feedback mode (OFB).

ECB and CBC are operate on cipher, whereas CFB and OFB are block cipher modes, which can be used as if they are working on stream cipher.



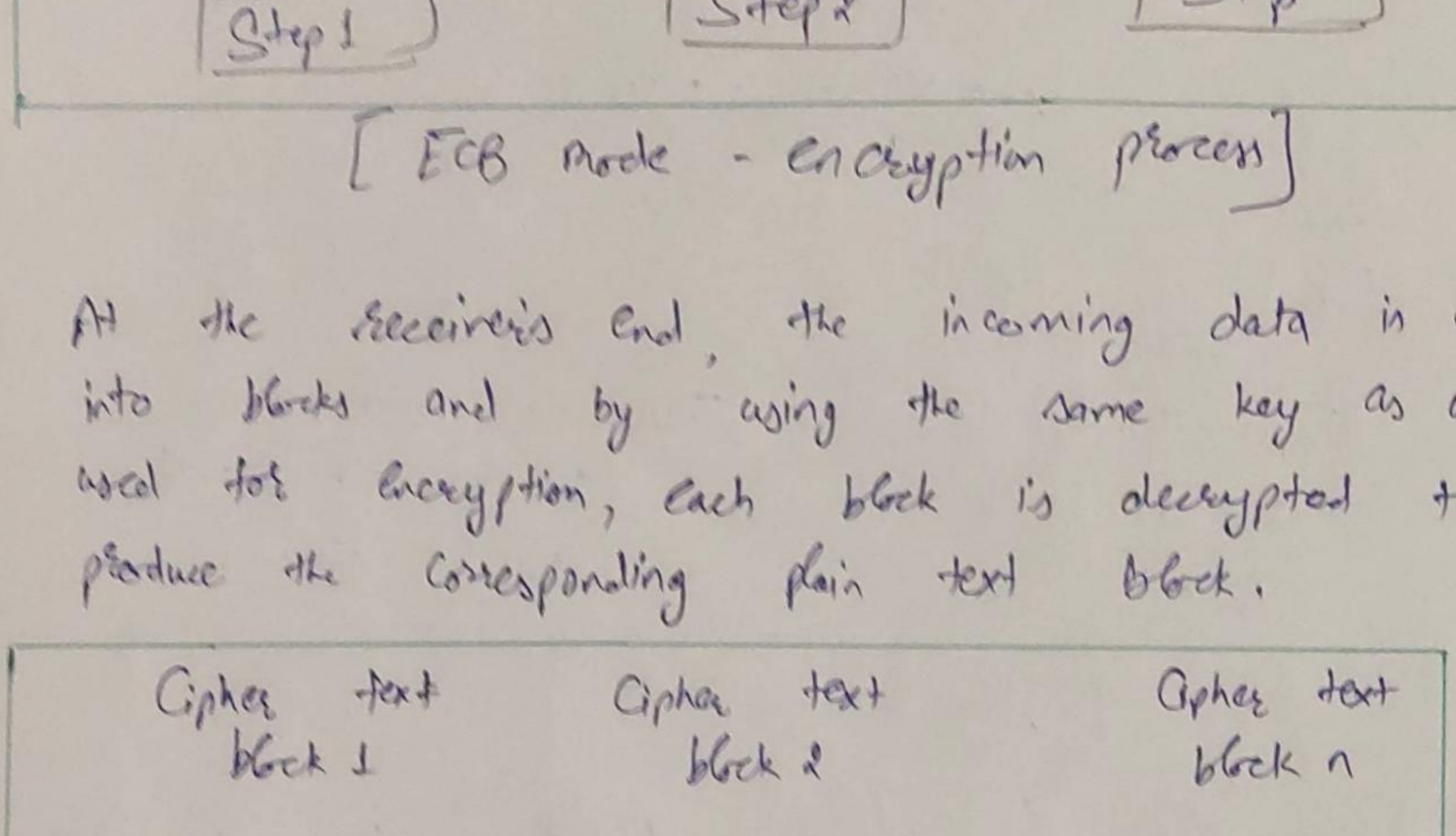
[Algorithm Modes]

Electronic Code Book (ECB) Mode : incoming plain text msg is divided into blocks of 64 bits each. Each such block is then encrypted independently of the other blocks. For all block in a msg., the same key is used for encryption.



[ECB Mode - encryption process]

At the receiver's end, the incoming data is divided into blocks and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block.



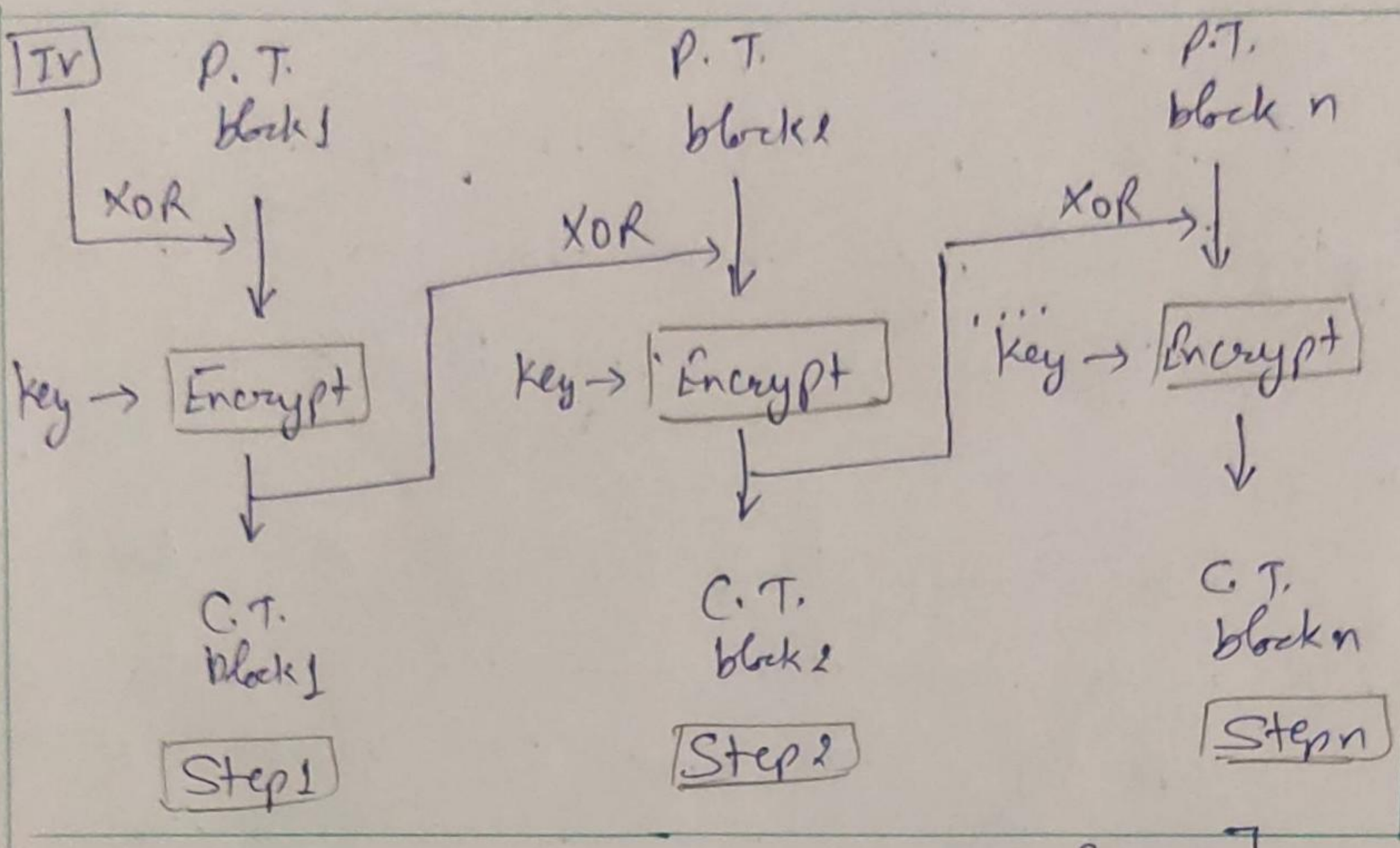
[ECB mode — decryption process]

~~Cipher~~ Single key is used for encrypting all the blocks of a msg. If a P.T. block replats in the original msg., the corresponding C.T. block will also repeat in the encrypted msg. Therefore, ECB is suitable only for ~~the~~ encrypting small msg., where the scope of the for repeating the same P.T. block is quite less.

Cipher Block Chaining Mode (CBC): If a block of P.T. occures more than once in the input, the corresponding C.T. block will also occur more than once in the output. To overcome this prob., The CBC modes ensures that even if a block of P.T. replats in the input, these two ~~Eid.~~ or (more) identical P.T. blocks yield totally diff cipher text blocks in the output. For this a feedback mechanism is used.

Chaining adds a feedback mechanism to a block Cipher. In CBC, the results of the encryption of the prevtous block are fed back into the encryption of current block. That is, each block is used to modify the encryption of next block.

Thus, each block of C.T. is dependent on the corresponding current input P.T. block, as well as all the previous P.T. blocks.



[CBC mode - Encryptim Process]

1) The first step receives two inputs: the first block of P.T. and a random block of text, called as Intialization Vector (IV).

a) The IV has no special meaning: It is simply used to make each msg. unique. Since the value of IV is randomly generated. IV helps in making the C.T. somewhat unique or at least quite diff from all other C.T. in a diff. msg.
It is not mandatory to keep IV secret - it can be known to everybody.

b) The first block of C.T. and IV are combined (15) using XOR and then encrypted using a key to produce the fiewst C.T. block. The fiest C.T. block is then provided as a feedback to the next P.T. block.

2) In the second step, the second P.T. block is XORed with the output of step 1, ie. the first C.T. block. It is then encrypted with the same key, as used in step 1. This produces C.T. block 2.

3) In the third step, the third P.T. block in XORed with the output of step 2, ie. the second C.T. block. It is encrypted with the same key, as used in step 1.

4) This process continues for all the remaining P.T. block. However, the same key is used for encryption of all P.T. blocks.

Note: IV is used only in the first P.T. block. However, the same key is used for encryption of all P.T blocks.

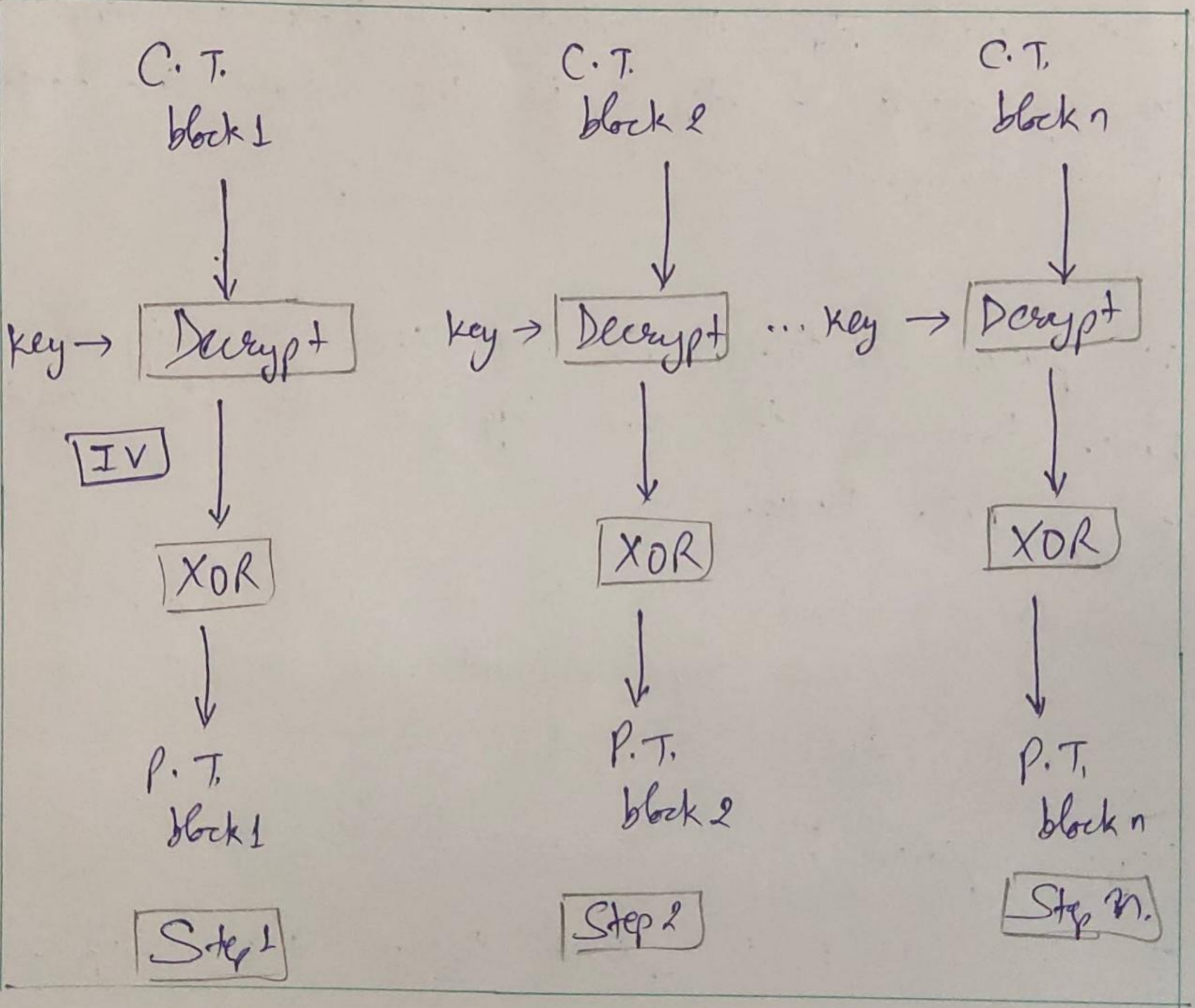
Decryption process:

1) The C.T. block 1 is passed through the decryption algo using the same key, which was used during

the encryption process for all the P.T. blcks. The output of this step is then XoRed with the IV. This process yields P.T. block 1.

2) In step 2, The C.T. block 2 is decrypted and its output is XoRed with C.T. block 1, which yields P.T. block 2.

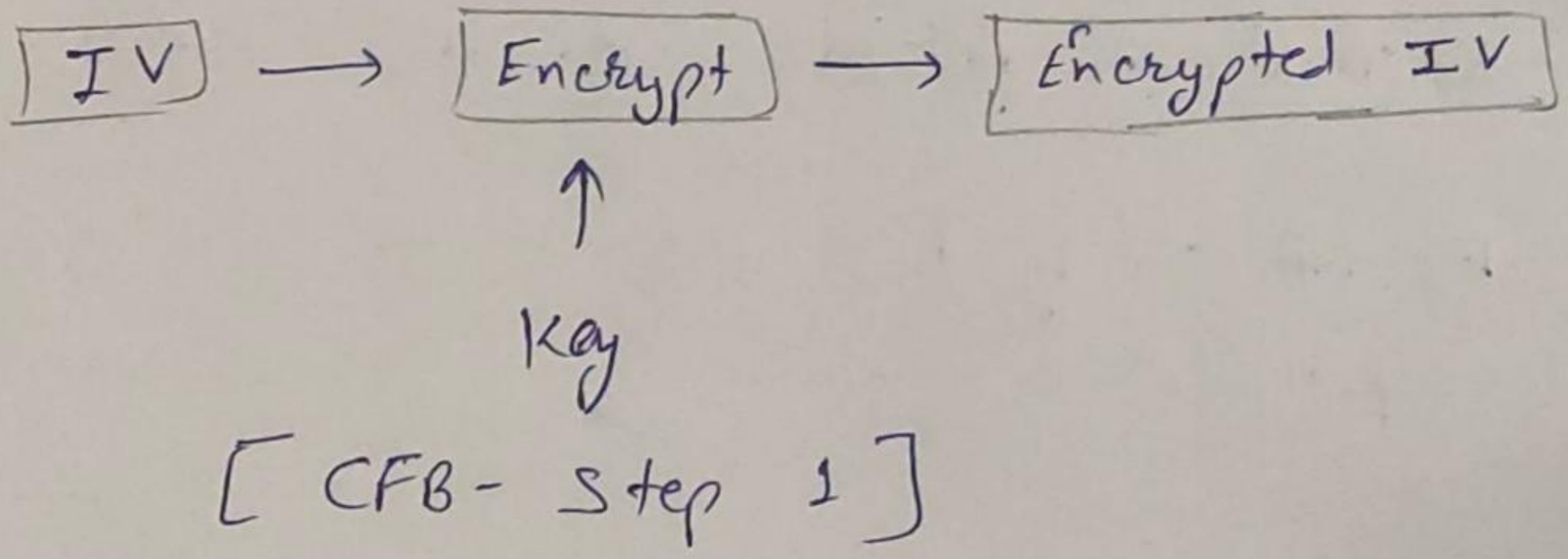
3) This process continues for all the C.T. blocks in the encrypted msg.



[CBC mode - decryption process]

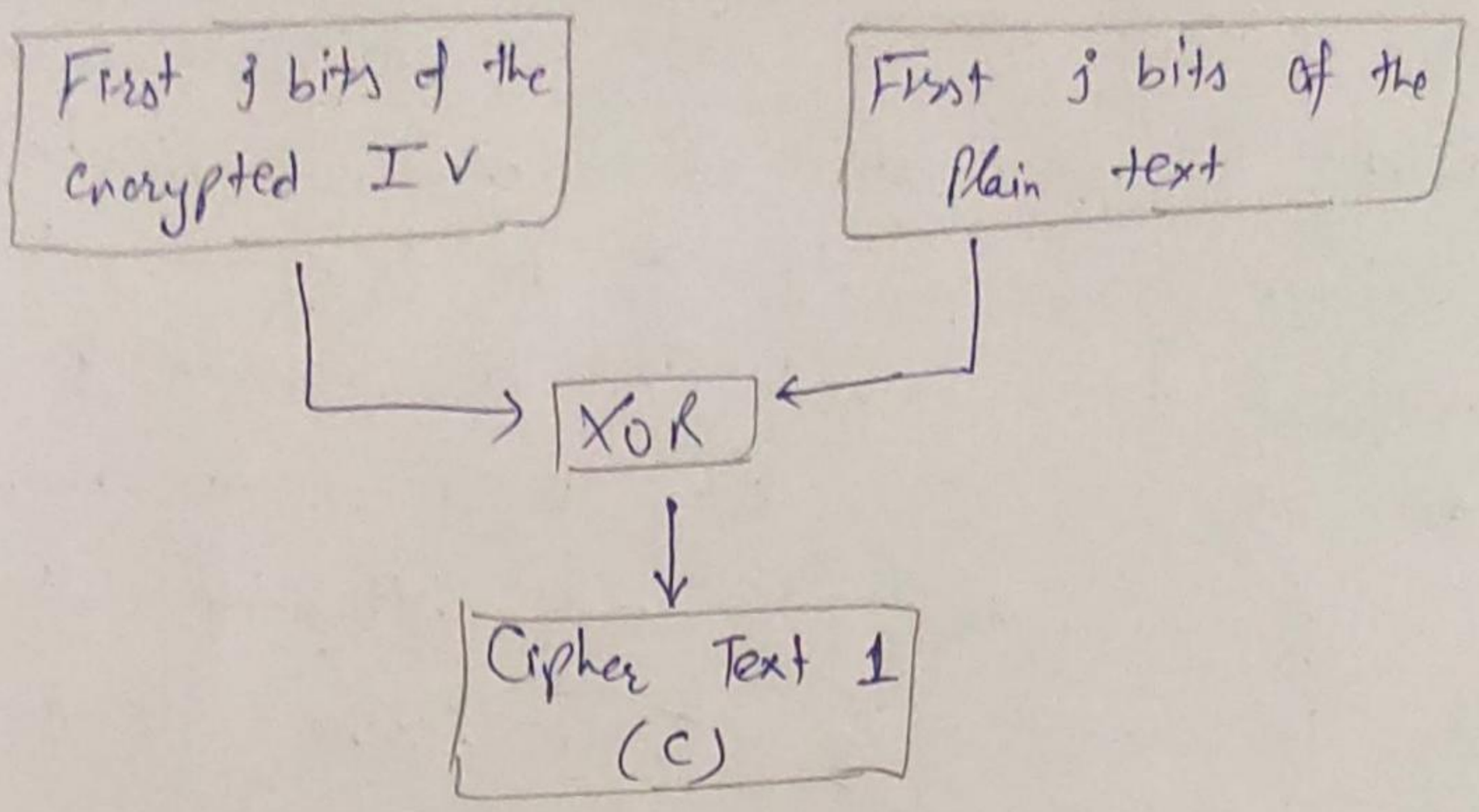
Cipher Feedback (CFB) Mode: Security is also required in app. that are character - oriented. For instance, an operator can be typing keystrokes at a terminal, which need to be immediately transmitted across the Comm' link in a secure manner, i.e. by using encryption. In such situation, stream cipher must be used. The Cipher Feedback (CFB) mode is useful in such cases.

Step 1: Like CBC, a 64-bit Initialization Vector (IV) is used in case of CFB mode. The IV is kept in a shift register. It is encrypted in the first step to produce a corresponding 64 bit IV Cipher text.



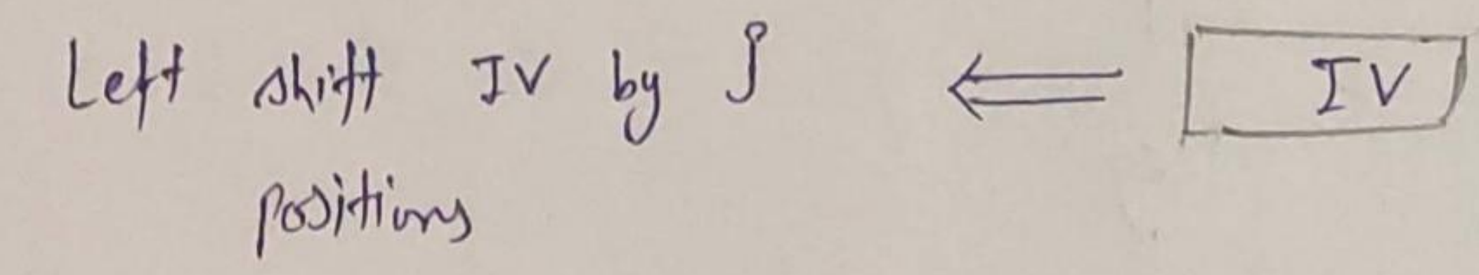
[CFB - Step 1]

Step 2: Now, the leftmost (i.e. the most significant) j bits of the encrypted IV are XORed with the first j bits of the P.T. This produces the first portion of C.T. (C) and C is transmitted to the receiver.



[CFB - step 2]

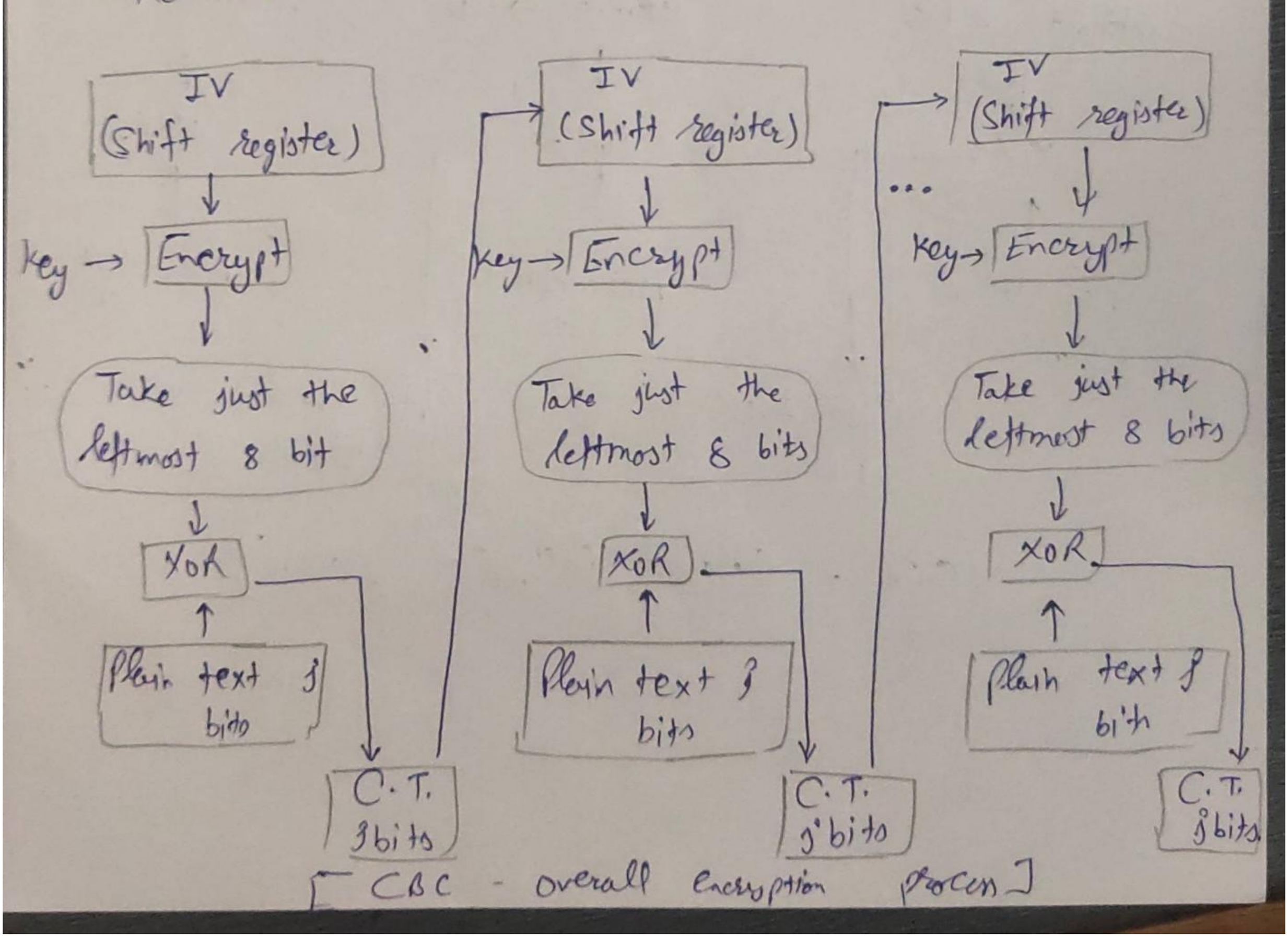
Step 3: Now, the bits of IV (i.e. the contents of the shift register containing IV) are shifted left by j positions. Thus, the Eightmost j position of the shift register now contain unpredictable data. These Eightmost j positions are now filled with C.



[CFB - Step 3]

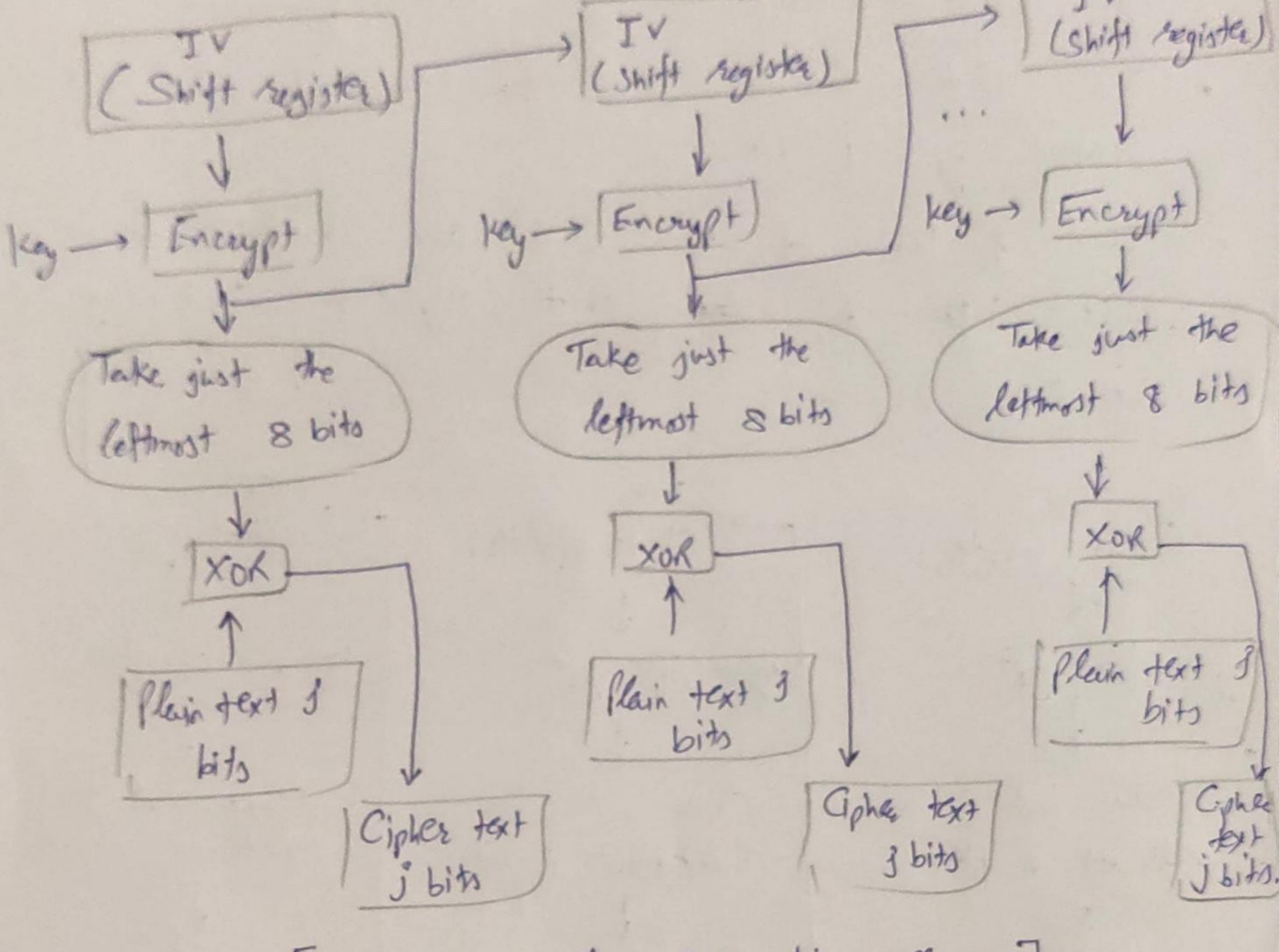
Step 4: Now, steps 1 through 3 continue until all the P.T. units are encrypted. That is, the following steps replat:

- IV is Encrypted.
- The leftmost j bits resulting from this encryption process are XORed with the next j bits of the P.T.
- The resulting C.T. portion (i.e the next j bits of C.T.) is sent to the receiver.
- The shift register containing the IV is left shifted by j bits.
- The j bits of the C.T. are inserted from right into the ~~left~~ shift register containing the I.V.



[CBC - overall Encryption process]

Output Feedback (OFB) Mode : The OFB is similar to the CFB. The only difference is that in case of CFB, the C.T. is fed into the next stage of encryption process. But in case of OFB, the output of IV Encryption process is fed into the next stage of Encryption process.



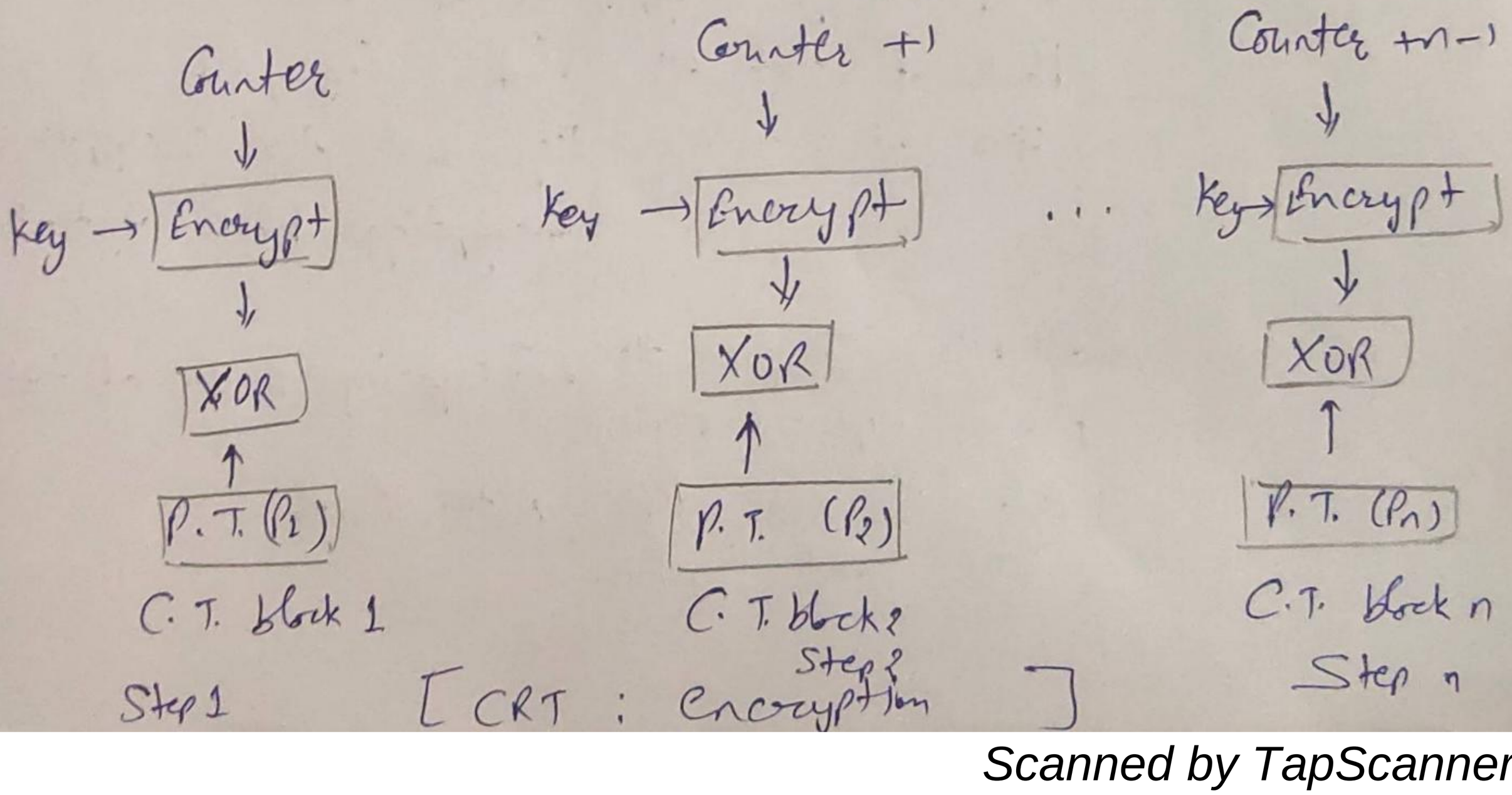
[OFB - Overall encryption process]

Counter (CTR) Mode: The CRT is quite similar to the OFB mode, with one variation. It uses sequence No. called as counters as the inputs to the algo. After each block is encrypted, to fill the register, the next counter value is used.

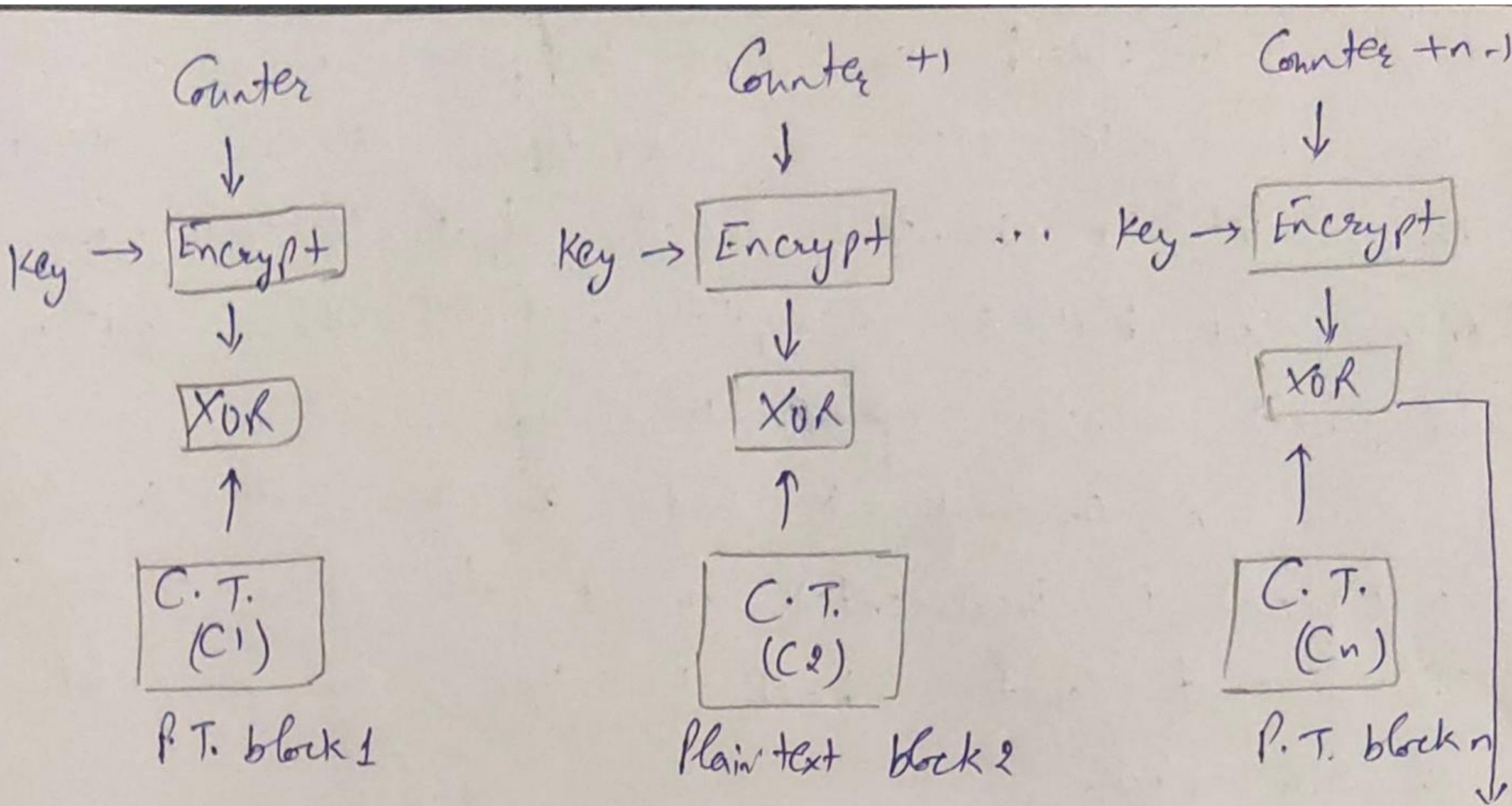
A constant is used as the initial counter value and is incremented by 1 for every iteration. The size of counter block is the same as that of P.T. block.

For encryption, the counter is encrypted and then XORed with P.T. block to get the C.T.

No chaining process is used. On decryption, the same sequence of counter is used. Here, each encrypted counter is XORed with the corresponding C.T. block to obtain the original P.T. block.



[CRT : Encryption]



[CRT : decryption process]

Algorithm Modes : Details & Usage :

Algorithm mode	Details	Usage
ECB	The same key independently encrypts blocks of text, 64 bits at a time	Transmitting a single value in a secure fashion (eg. password or key used for encryption) ~~encrypting block of text~~
CBC	64 bits C.T. from the previous step and 64 bits P.T. of the next step are XORed together.	• Encrypting blocks of text • Authentication.

CFB	K bits of randomized C.T. from the previous step and K bits P.T. of the next step are XORed together.	• Transmitting encrypted stream of data • Authentication
OFB	Similar to CFB except that input to the encryption step is the preceeding DES output.	• Transmitting encrypted stream of data
CTR	A counter and P.T. block are encrypted together, after which the counter is incremented.	• Block - oriented transmission. • App. needing high speed.