CNS UNIT-2 - Cns

Cryptography & Network Security (Jawaharlal Nehru Technological University, Hyderabad)

UNIT – II :-

→ An algorithm type defines what size be encrypted in each step of an algorithms.

→ The generation of cipher text from plain text is done in two basic ways i.e. stream ciphers & block ciphers.

→ Stream-cipher technique involves the encryption of one plain-text bit at a time. The decryption also happens one bit at a time.

→ Block-cipher technique involves encryption of one block of plaintext at a time. Decryption also takes one block of encrypted text at a time.

Block-cipher principles :-

→ A block cipher operates on blocks of data.

→ Algorithm breaks the plaintext into blocks and operates on each block independently.

→ Usually blocks are 8 or 16 bytes long.

→ Security of blocks ciphers depends on the design of the encryption function.

→ Software implementations of blocks ciphers run faster than software implementations of stream ciphers.

→ Errors in transmitting one block generally do not affect other blocks.

Advantages :-

1. High diffusion
2. Immunity to insertations of symbols.

Disadvantages :-

1. Slowness of encryption.
2. Error propagation.

Confusion and Diffusion :-

Diffusion and confusion were introduced by claude Shannon to capture the two basic building blocks for any cryptographic system.

	Diffusion		Confusion
1	Diffusion hides the relation blw the ciphertext and the plaintext	1	confusion hides the relationship blw the ciphertext and key.
2	If a single symbol in the plaintext is changed, several or all symbols in the ciphertext will also be changed.	2	If a single bit in the key changed most or all bits in the ciphertext will also be changed.
3	In diffusion, the statistical structure of the plaintext is dissipated into long range statistics of the cipher text. This is achieved by permutation	3.	In confusion, the relationship blw the statistics of the cipher text & the value of the encryption key is made complex. It is achieved by substitution.

Feistel Cipher :-

→ Feistel proposed classical classical Feistal block cipher.

→ In this block cipher key length is K bits and block length of n bits allowing a total of 2^K or $2^n!$ possible transformation The plaintext block is divided into two halves ie Left (L_0) and right (R_0).

→ Feistel proposed the use of a cipher that alternates substitution and permutations.

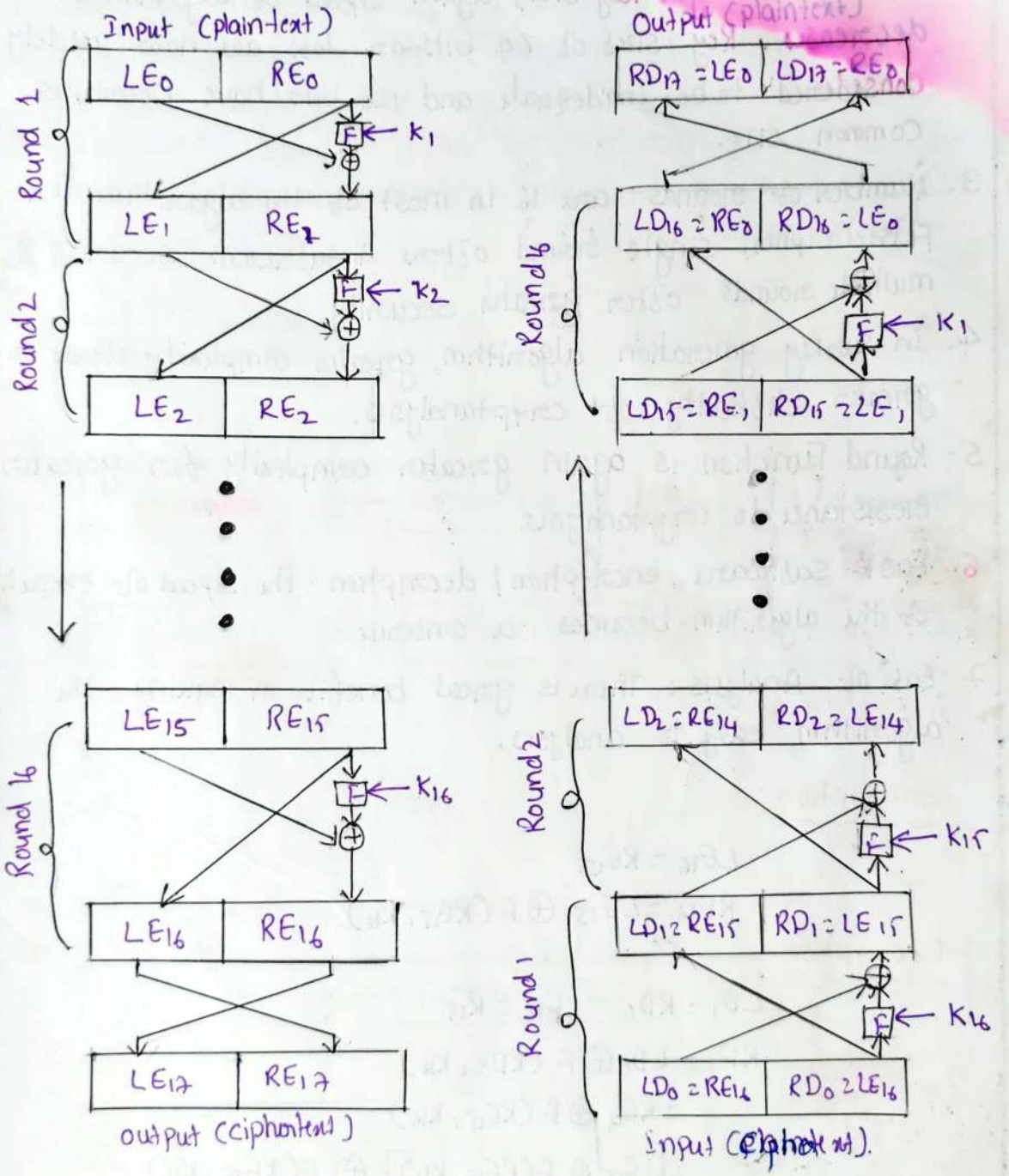
1. Substitution :

Each plain text or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.

2. Permutation :-

A sequence of plaintext elements is replaced by a permutations of that sequence. That is, no elements are added or deleted or

replaced in the sequence, rather the order in which the appear in the sequence is changed.



Feistel Encryption and Decryption (16 Rounds).

1. Security depends upon the block size. Larger block size gives greater security but encryption / decryption speed is reduced normal. Block size is 64-bit.

2. Greater security is achieved by using longer key size. Because of longer key size, again speed of algorithm decreases. Key sizes of 64 bits or less are now widely considered to be inadequate and 128 bits have become a common size.

3. Number of rounds are 16 in most of the algorithm. In Feistel ciphers, single round offers insufficient security & multiple rounds offer greater security.

4. In subkey generation algorithm, greater complexity leads to greater difficulty of cryptanalysis.

5. Round Function is again greater complexity for greater resistance to cryptanalysis.

6. Fast software encryption / decryption : The speed of execution of the algorithm becomes a concern.

7. Ease of Analysis : There is great benefit in making the algorithm easy to analysers.

Encryption :-

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16}).$$

Decryption :-

$$LD_1 = RD_0 = LE_{16} = R_{15}$$
$$RD_1 = LD_0 \oplus F(RD_0, K_{16})$$
$$= RE_{16} \oplus F(RE_{15}, K_{16})$$
$$= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$$

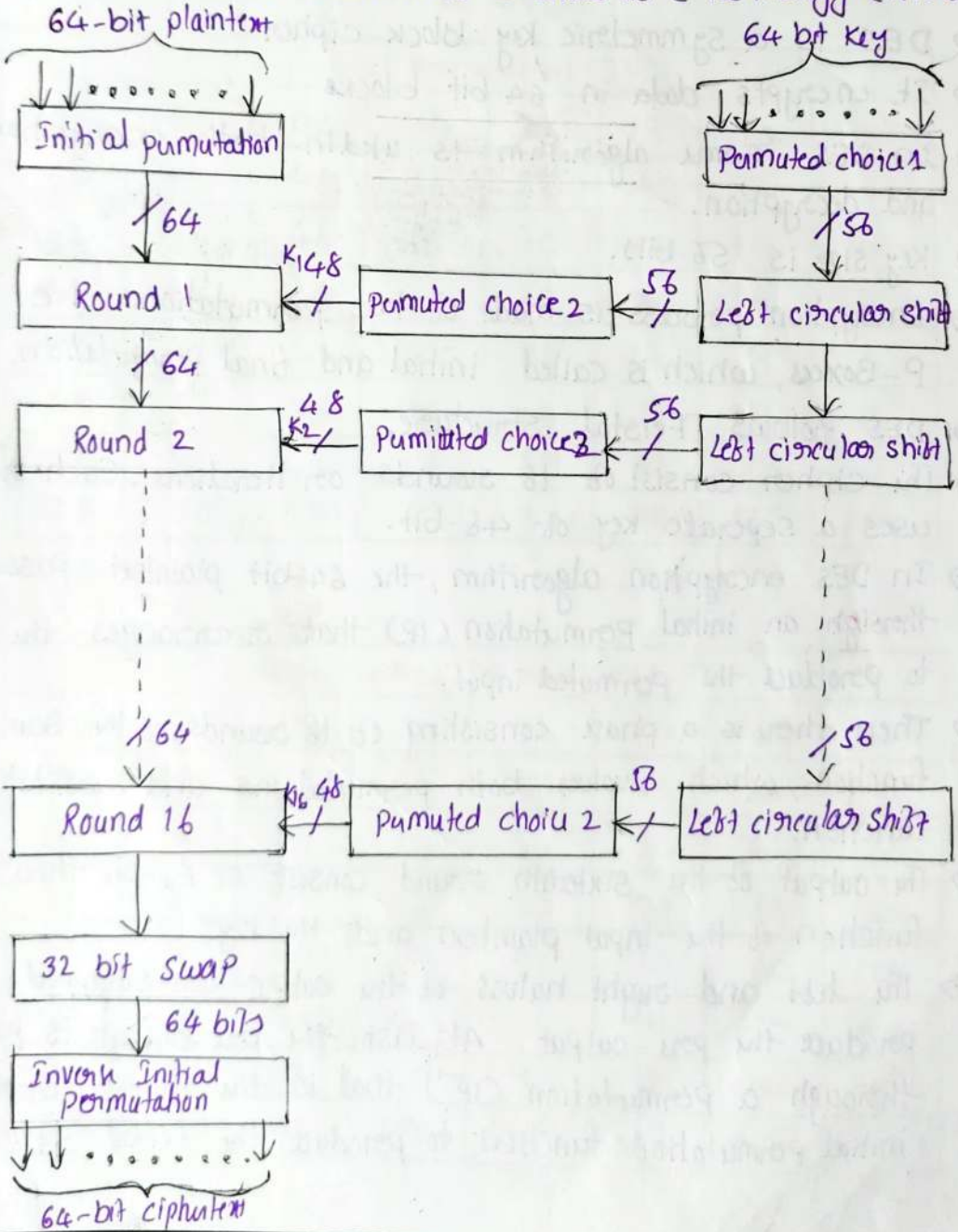
ith Iteration :

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

Data Encryption Standard :-

→ Until the introduction of the Advanced Encryption Standard (AES) in 2001, the Data Encryption Standard (DES) was the most widely used encryption scheme.

→ DES was issued in 1977 by National Bureau of Standards now the National Institute of Standards & Technology (NIST).



General Depiction of DES Encryption Algorithm.

In the encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits length (Actually, the function expects a 64-bit key as input. But only 56 of these bits are ever used, the other 8 bits are parity bits.

→ DES is a Symmetric key block cipher
→ It encrypts data in 64 bit blocks
→ In DES same algorithm is used in both encryption and decryption.
→ Key size is 56 bits.
→ Encryption process is made of two permutations i.e. P-Boxes, which is called initial and final permutation.
→ DES follows Feistal structure.
→ The cipher consist of 16 rounds or iterations. Each rounds uses a seperate key of 48-bit.
→ In DES encryption algorithm, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.
→ Then there is a phase consisting of 16 rounds of the same function, which involves both permutations and substitution function.
→ The output of the sixteenth round consist of 64-bit there are a function of the input plaintext and the key.
→ The left and right halves of the output are swapped to produce the pre-output. At last, the pre-output is passed through a permutation (IP^{-1}) that is the inverse of the initial permutation function to produce the 64-bit ciphertext.

Initial permutation :-

→ The Below table shows the initial permutation and its inverse. The input to a table consist of 64-bit numbered from 1 to 64.

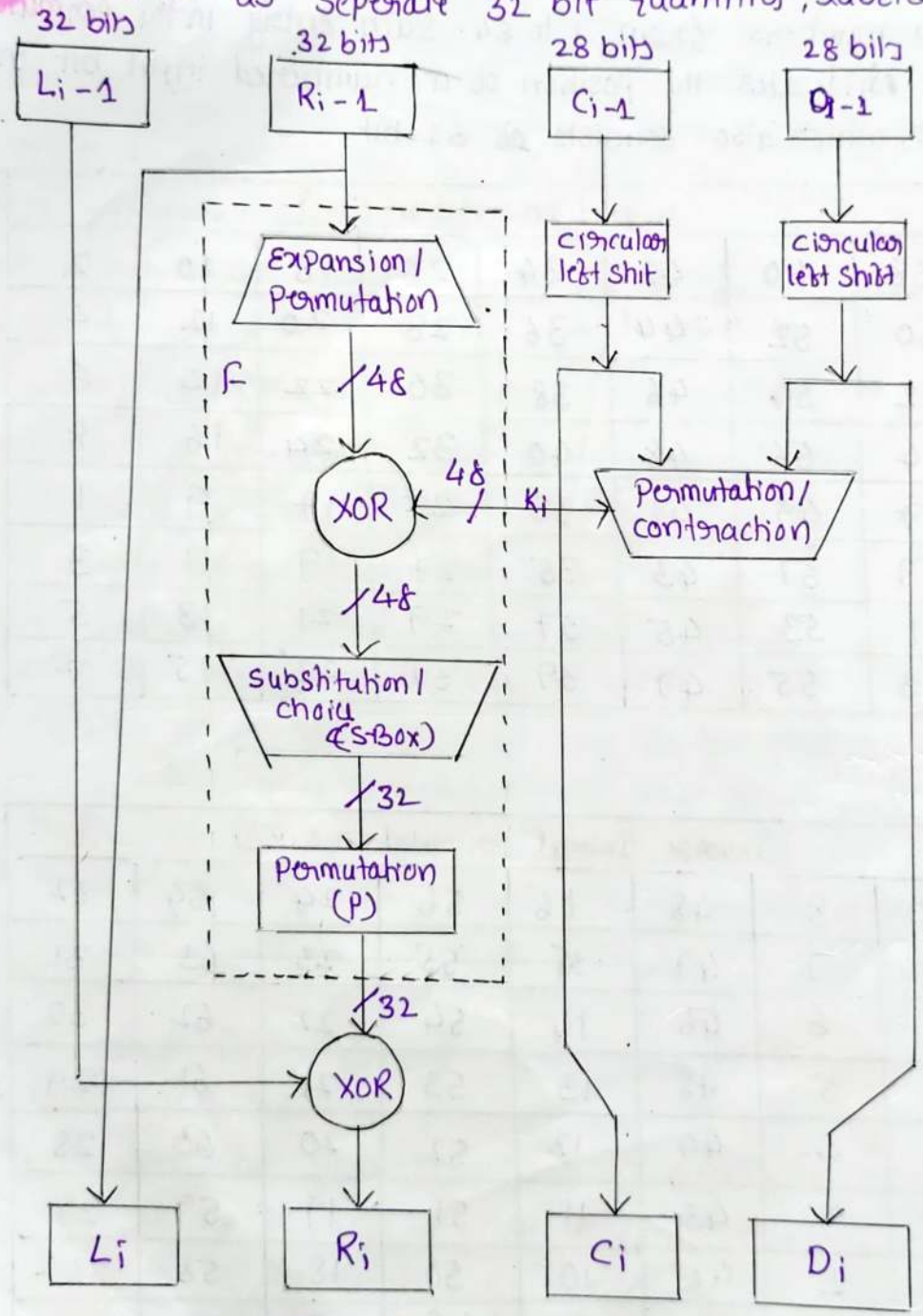
→ The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64-bit.

Initial permutation (IP)							
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

Inverse Initial permutation (IP^{-1})							
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

Details of single Round :-

The below diagram shows single round of DES algorithm. The left and right halves of each 64-bit intermediate value are treated as seperate 32 bit quantities, labeled L and R.



Single round of DES Algorithm.

→ The overall processing at each round can be summarised in the following formulas :

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \times F(R_{i-1}, K_i)$$

→ The left output (L_i) is simply copy of the right input (R_{i-1}). The right output (R_i) is the XOR of left input (L_{i-1}) and right input (R_{i-1}) and key for this stage is K_i. In this stage, the substitution and permutation both functions are used.

Permutation choice 1 (NO 8th bits)

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

Permutation choice 2 (contraction)

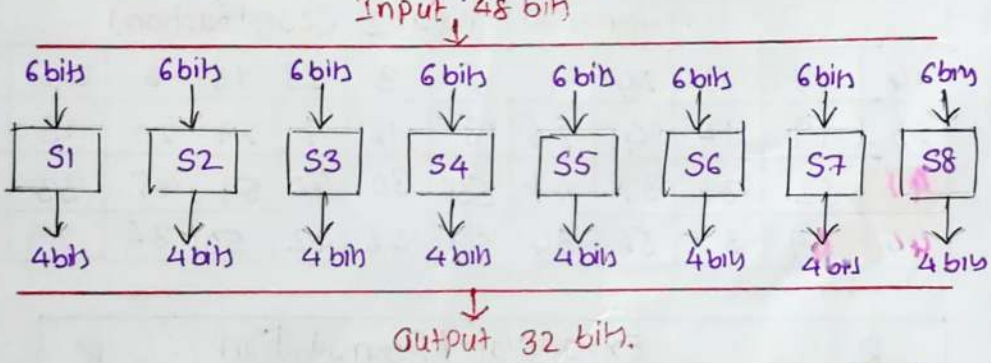
14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

R		Expansion / Permutation			R
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	30	31	32	01

Left circular shift								
Round	1	2	3	4	5	6	7	8
Bits shifted	1	1	2	2	2	2	2	2
Round	9	10	11	12	13	14	15	16
Bits shifted	1	2	2	2	2	2	2	1

Permutation function (P)							
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

→ In function F consist of set of Eight S-Boxes, each of which accepts 6 bits as input and produces 4 bits as output.



→ First and Last bits of input to box S_i form a 2 bit binary number to select one of four substitutions defined by the four rows in the table for S_i. Two bits can store any decimal number blw 0 and 3 (00, 01, 10, 11). This specifies the row number the middle four bits select one of the sixteen columns. (0000 — 1111).

F L
1 0 0 1 1 0

first+LAST → row number 10 - row → 2

Blw four bits → column number 0011 → column → 3

→ In S1 box row 2 and cloumn 11 value is 8 binary equivalent is 1000.

→ following tables gives the S box values for DES.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S1 0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
S1 1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
S1 2	4	1	14	8	13	6	2	11	15	12	9	9	3	10	5	0
S1 3	15	12	8	2	4	9	1	7	5	11	3	3	10	0	6	13

S2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

						11	1	10	9	3	14	5	0	12	7
					3	7	4	12	5	6	11	0	14	9	2
				9	12	14	2	0	6	10	13	15	3	5	8
		14	7	4	10	8	13	15	12	9	0	3	5	6	11

→ Same process implemented in each round. After 16th round output is 32 bits swap. then we get the 64 bit data. use invers initial pumutations IP^{-1} then we get the 64 bits cipher text

→ The decryption also same process inplace of 64-bits plain text it considered 64 bits cipher text.

Key Generation :-

→ 64-bit Key is used as input to the algorithm. The initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key.

→ From 56-bits key, a different 48-bit Sub Key is generated during each round using a process called as key transformation.

→ The resulting 56-bit key is then treated as two 28-bit quantities labeled C_0 and D_0. At each round C_{i-1} and D_{i-1} are seperately subjected to a circular left shift or rotation of 1 or 2-bits.

→ These shifted values serve as input to the next round. They also serve as input to permuted choice two, which produces a 48-bit output that serves as input to the function $F(R_{i-1}, K_i)$.

Advantages of DES :-

1. As 56 bit keys are used there are 70 quadrillion possible key values and hence a specific key can't be Indentified easi
2. As the length of the key is increased the security provided by the algorithm also increases.
3. The security of the DES algorithm resides in the key.

Disadvantages of DES :-

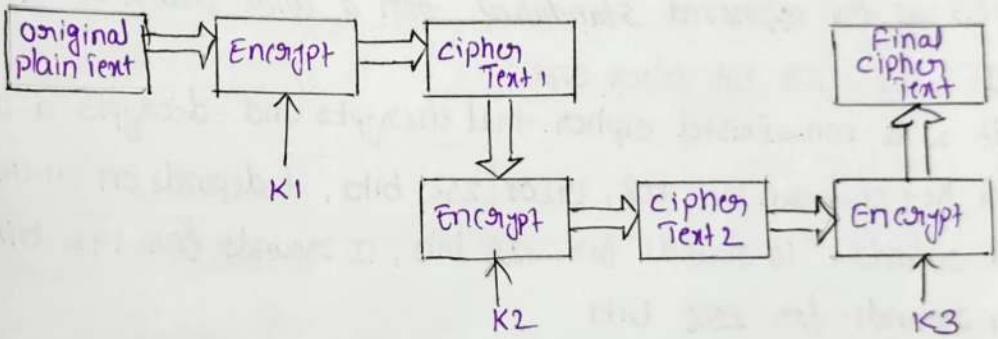
→ Both sender and reciver must have same key. The possibility that the key is intercepted.
→ The design of S-boxes makes it susceptible to linear cryptanalysis attack.
→ Brute force attack which DES cracks.
→ It has certain weak keys. 0000000 0000000, 0000000 ffffffff, ffffffff 0000000, ffffffff ffffffff.

Possible techniques for improving DES :-

→ Multiple enciphering with DES
→ Extending DES to 128-bit data paths and 112-bit keys
→ Extending the key expansion calculation.

Triple DES :-

→ In 1999, NIST issued a new version of its standard that indicated that DES should be used only for legacy systems and that triple DES (repeating DES algorithms three times) be used.



Triple DES with three keys.

→ It takes three 64 bit keys, for an overall key length of 192 bits.
→ The procedure for encryption is exactly the same as regular DES, but it is repeated three times. Hence the name triple DES.
→ Triple DES uses 2 or 3 keys.

data is encrypted with the first key (K1), decrypted with second key (K2) and finally encrypted again with third key (K3). or (K1)

→ Triple DES with three keys is used quite extensively in many products including PGP and S/MIME.

→ Brute force search impossible on triple DES.

→ cipher text is produced as

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

→ plain text is produced at decryption as

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

→ Triple DES runs three times slower than standard DES, but is much more secure if used properly.

Advanced Encryption Standard (AES) :-

→ The AES was published by the National Institute of Standards and Technology (NIST) in 2001.

→ AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications.

→ It uses 128 bit block size.

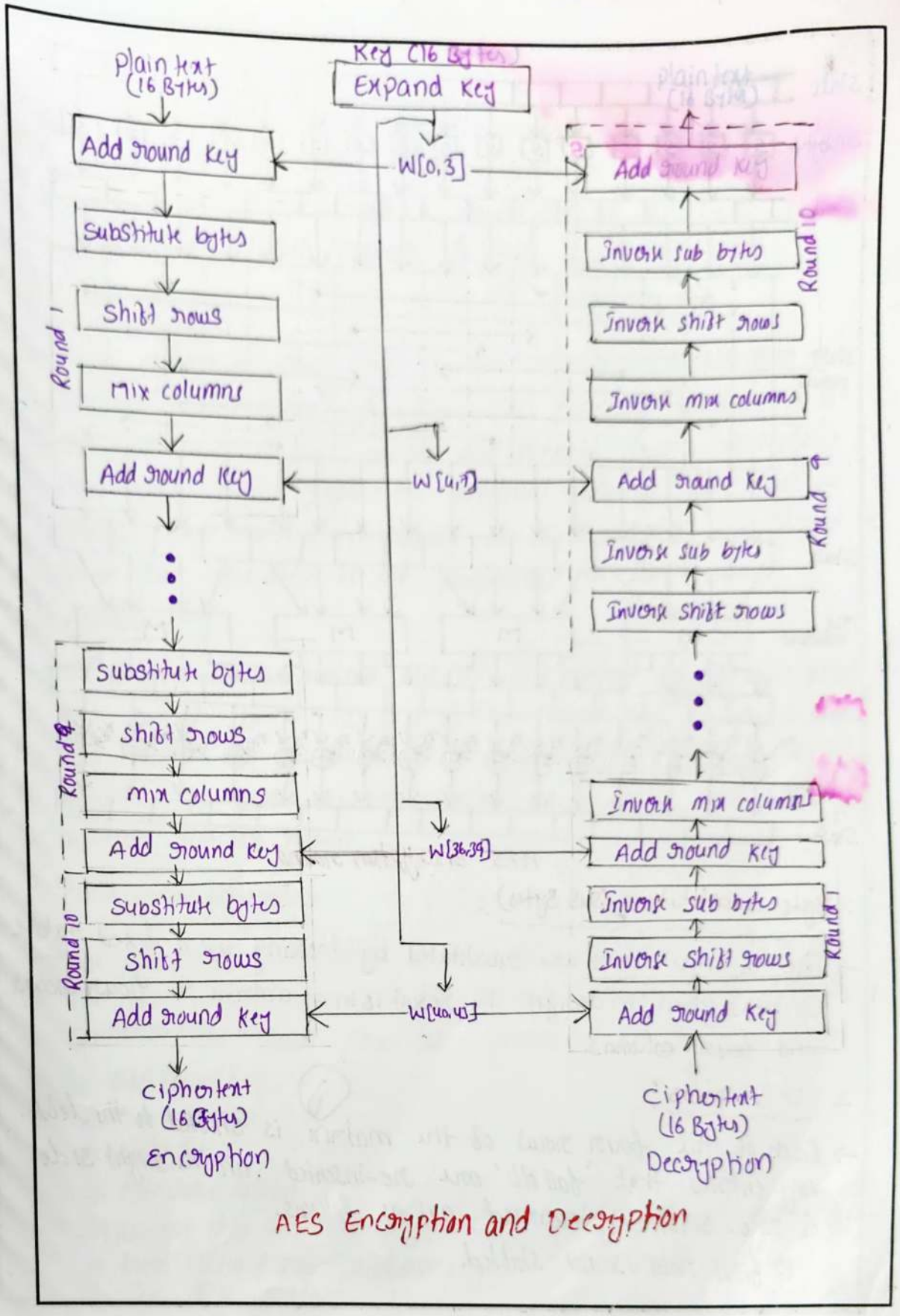
→ It is a non-Feistel cipher that encrypts and decrypts a data.

→ The key size can be 128, 192 or 256 bits, it depends on number of rounds. 10 rounds for 128 bits, 12 rounds for 192 bits 14 rounds for 256 bits

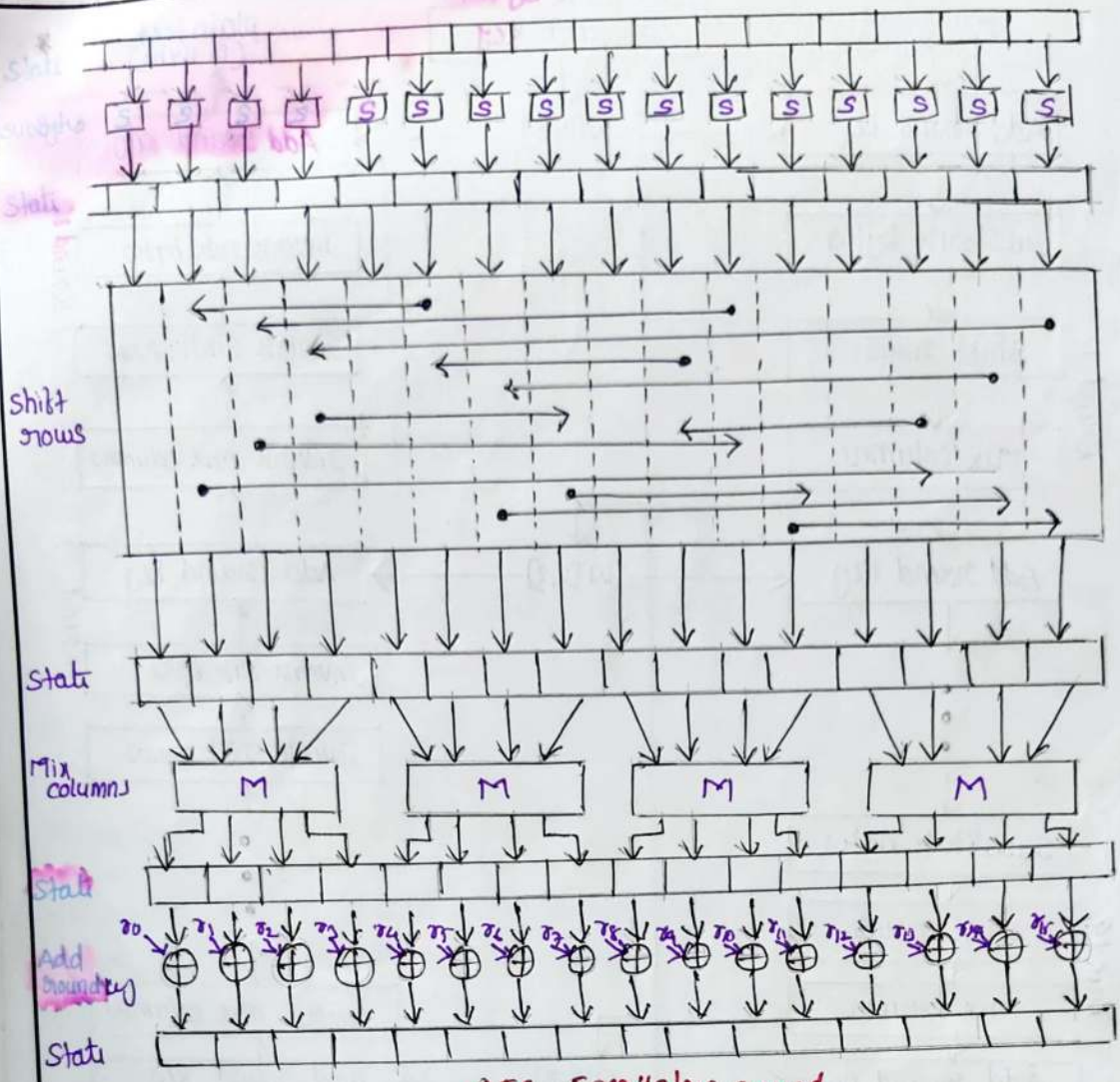
Encryption :-

→ In each round comprise of four subprocesses

1. Byte Substitution (Sub bytes)
2. Shift rows
3. Mix columns
4. Add Round Key



AES Encryption and Decryption



AES Encryption round.

1. Byte substitution (Sub Bytes) :-

→ The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

2. Shift rows :-

→ Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows.

1. first row is not shifted.

2. Second row is shifted one (byte) position to the left.
3. Third row is shifted two positions to the left.
4. Fourth row is shifted three positions to the left.
5. The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

3. Mix columns :-

→ Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in last round.

4. AddRoundKey :-

→ The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption process :

→ The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consist of the four processes conducted in reverse order.

1. Add Round Key
2. Mix columns
3. Shift Rows
4. Byte Substitution.

→ In present day cryptography, AES is widely adopted and supported in both H/w & S/w. Additionally AES has built in flexibility of the Key length.

Blow Fish :-

→ Blow Fish is a symmetric block cipher developed by Bruce Schneier in 1993. It is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable length key, from 32 bits to 448 bits.

Features :-

1. Fast :-
→ Blow fish encryption rate on 32-bit microprocessors is 26 clock cycles per byte.
2. Compact :-
→ Blowfish can execute in less than 5 KB memory
3. Simple :-
→ Blowfish uses only primitive operations such as Addition, XOR.
4. Secure :-
→ Blowfish has a variable key length up to maximum of 448 bits long, making it both flexible & secure.

Operation :-

→ Blow fish encrypts 64-bit blocks with a variable length key. It contains two parts

a. Subkey generation :-

→ This process converts the key up to 448 bits long to subkeys totaling 4168 bits. The important aspect of subkey generation process step by step.

i. Blow fish makes use of a very large number of subkeys. i.e. 32 to 448 bits other words 1 to 14 words.

K1, K2 ... Kn where $1 \leq n \leq 14$.

ii. We then have the concept of a P-array consisting of 18 32-bit sub keys

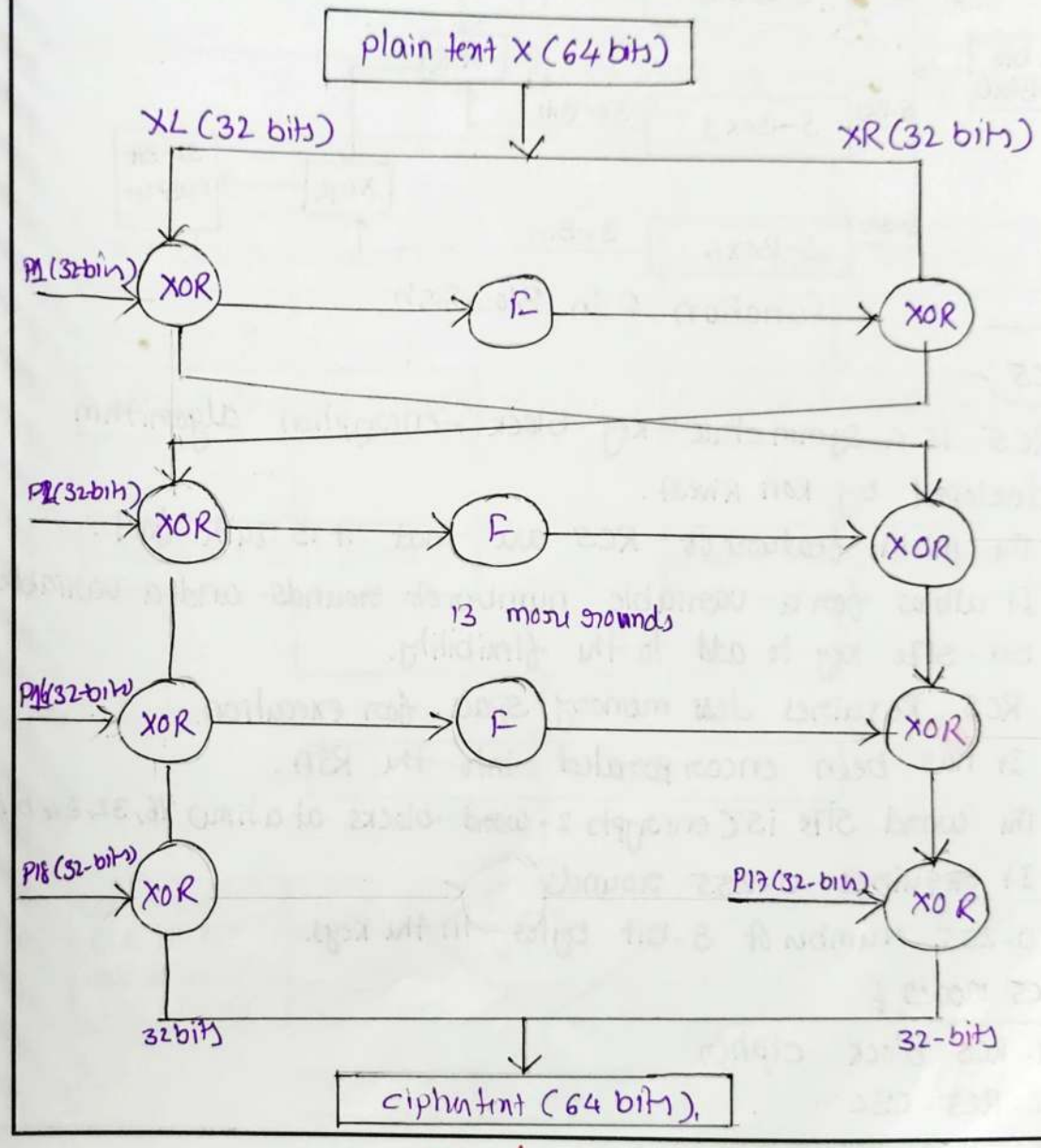
$P_1, P_2 \ldots P_{18}$.

iii. four S-boxes each containing 256 32-bit sub entries

S1, 0, S1, 1 ... S1, 255 S3, 0, S3, 1, ... S3, 255

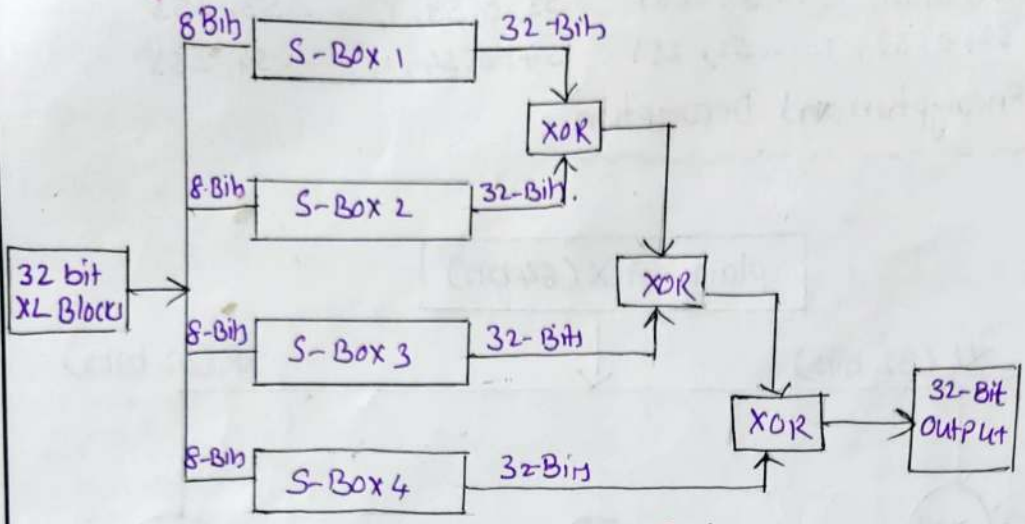
S2, 0, S2, 1 .... S2, 255 S4, 0, S4, 1, ... S4, 255.

b. Data Encryption and Decryption :-



Blowfish encryption

b. Compute $F[a,b,c,d] = ((S1, a+S2, b) \text{ XOR } S3, c) + S4, d.$



function F in BlowFish.

RC5 :-

→ RC5 is a symmetric-key block-encryption algorithm developed by Ron Rivest.

→ The main feature of RC5 are that it is quite fast.

→ It allows for a variable number of rounds and a variable bit-size key to add to the flexibility.

→ RC5 Requires less memory space for execution.

→ It has been encorporated into the RSA.

→ The word size is (encrypts 2-word blocks at a time) 16, 32, 64 bits.

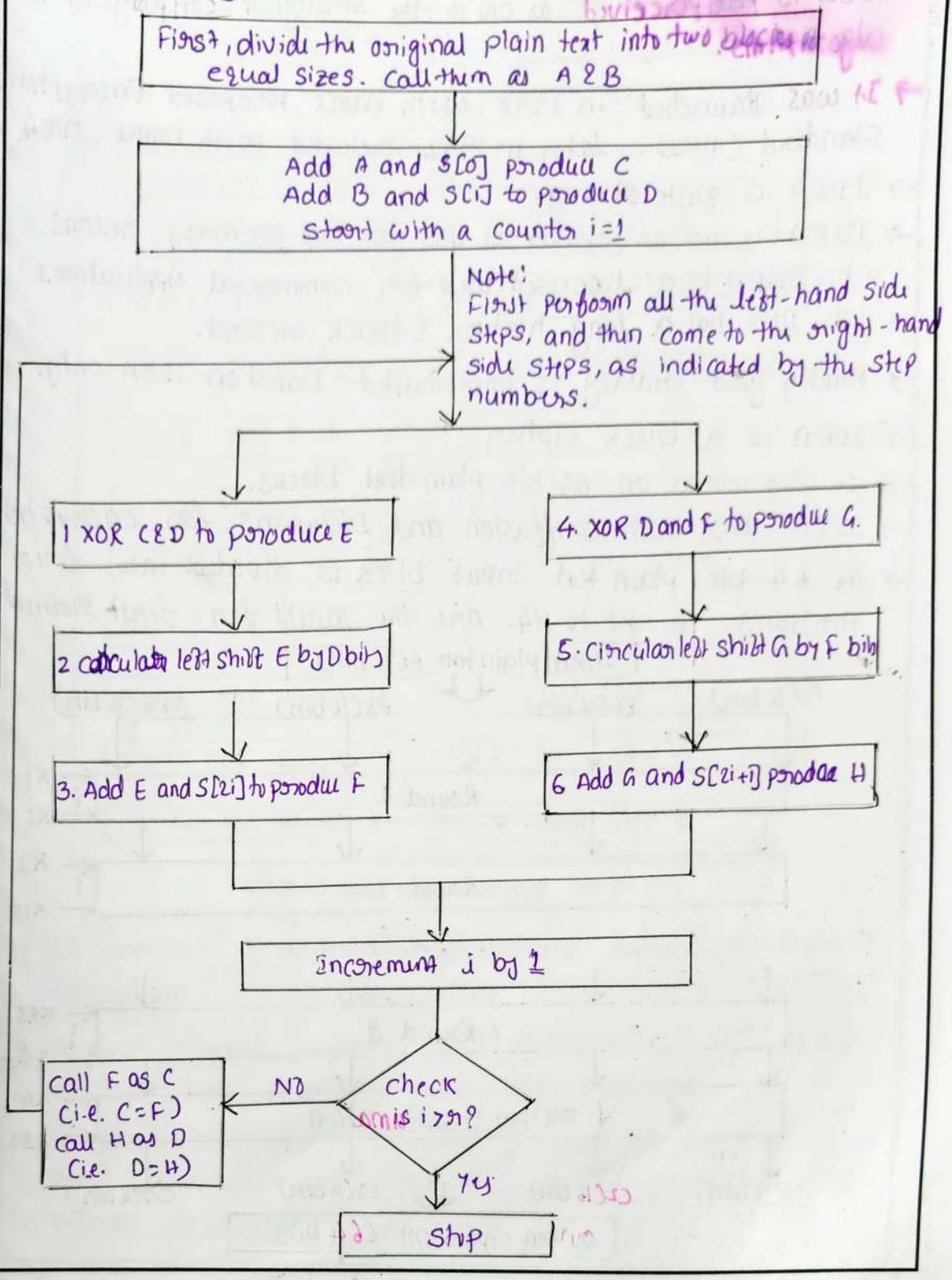
→ It requires 0-255 rounds

→ 0-255 Number of 8-bit bytes in the keys.

RC5 Modes :

1. RC5 block cipher
2. RC5-CBC

3. RC5-CBC-Pad
4. RC5-CTS.



Encryption using RC5

International Data Encryption Algorithm (IDEA) :-

IDEA is perceived as one of the strongest cryptographic algorithms out there.

→ It was launched in 1990 with name proposed Encryption Standard (PES). Later in 1992 released with name, IDEA.

→ IDEA is quite strong.

→ IDEA is not as popular as DES for two primary reasons
1. IDEA is a licensed used for commercial Applications.
2. DES has a long history & track record.

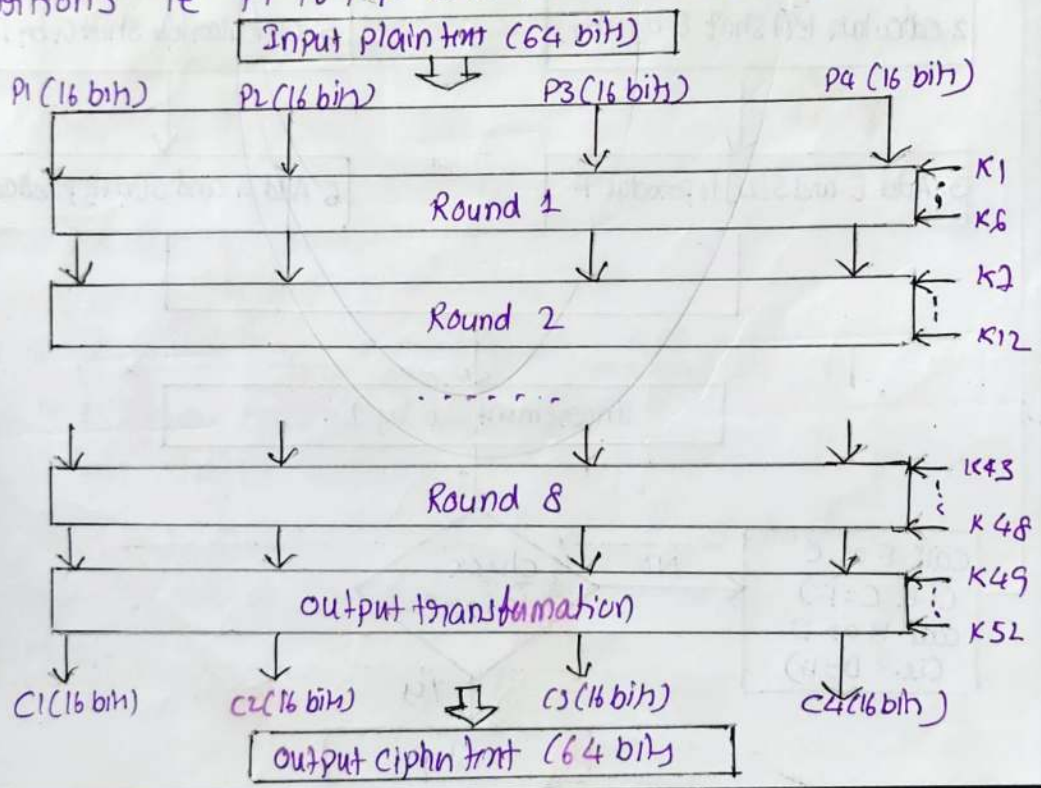
→ Pretty good privacy is implemented based on IDEA only.

→ IDEA is a block cipher

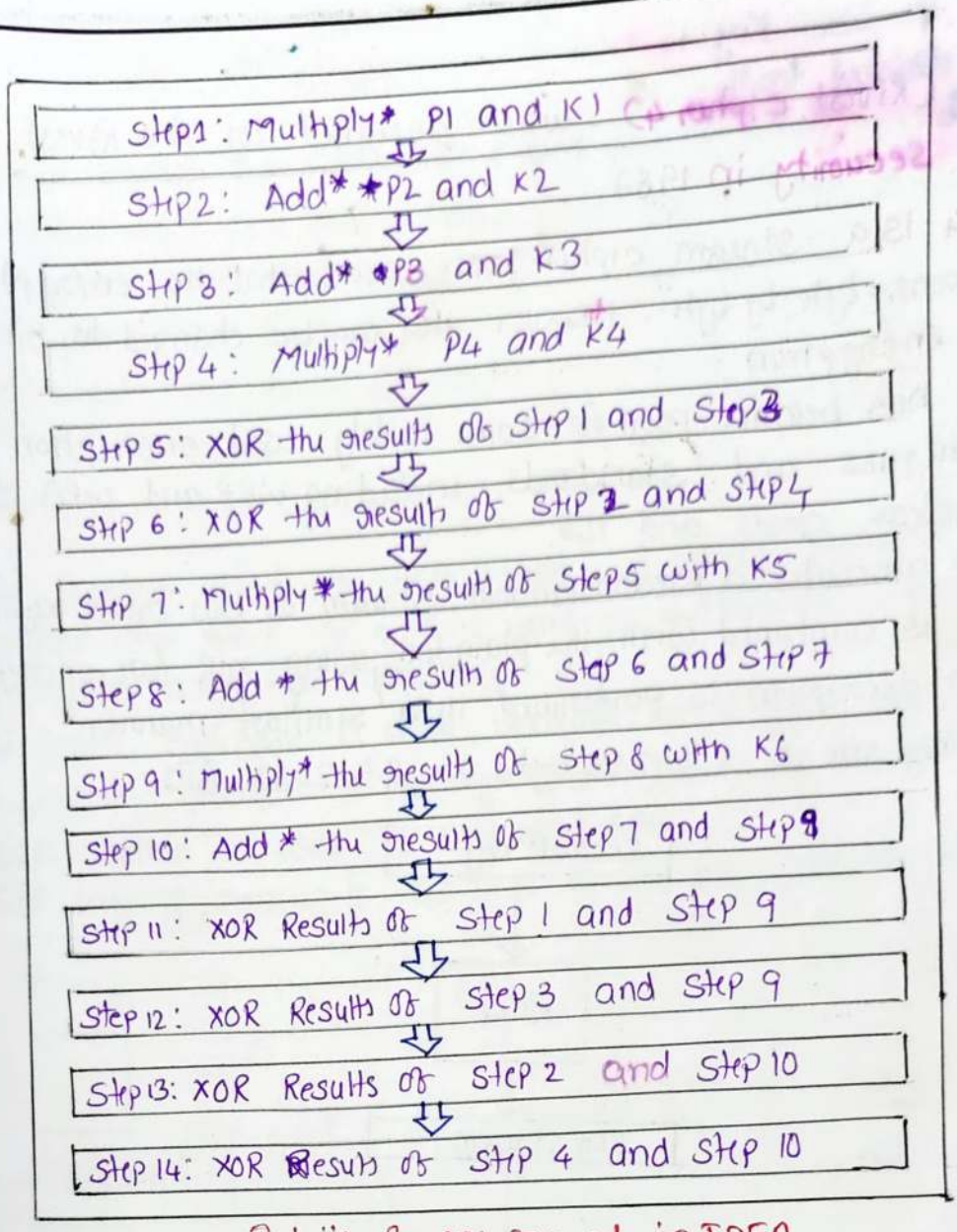
→ It also works on 64-bit plain-text blocks.

→ IDEA uses both confusion and Diffusions. for encryption.

→ The 64-bit plain text input block is divided into four portion's ie P1 to P4. are the input's for first round.



→ IDEA uses 128-bit key which is double than key size of DES.



Details of one round in IDEA

→ It consist 8 Rounds. Each round involves a series of operations on the four data blocks using six Keys.

→ In Each round perform lot of mathematical actions i.e. multiplication, Additions and XOR operations.

→ The decryption process is exactly same as the encryption process. There are some alterations in the generation and perform of subkeys.

RC 4 :-

RC4 (Rivest cipher 4) was designed by Ron Rivest of RSA security in 1987.

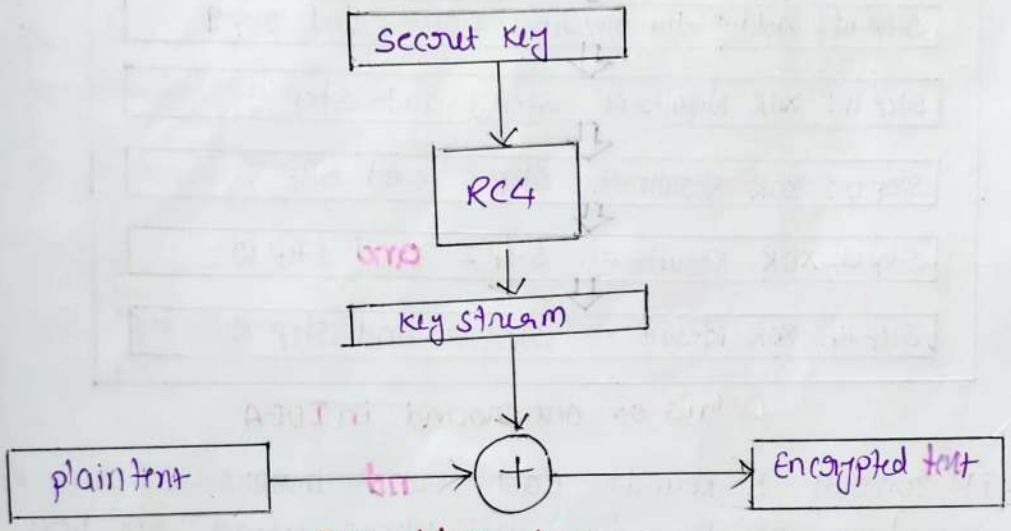
→ RC4 is a stream ciphus. This means that the encryption happens byte-by-byte. However, this can be changed to bit-to-bit encryption.

→ RC4 has become part of some widely used encryption techniques and standards including WEP and WPA for wireless cards and TLS.

→ RC4 generates a pseddorandom stream of bits called KeyStream. This is combined with the plain text using XOR for encryption.

→ Even decryption is performed in a similar manner.

→ The Key size is 1 to 256 bytes or 8 to 2048 Bits.



RC4 block diagram

→ cryptographically very strong and easy to implement

→ RC4 consist 2 parts

1. Key scheduling Algorithm (KSA) : Generate State Array
2. Pseduo-Random Generation Algorithm : Generate Key Stream, XOR Key Stream with the data to generated encrypted Stream

Block cipher operations :-

→ Block cipher consist five modes of operations

1. Electronic Code Book (ECB)
2. Cipher Block chaining (CBC)
3. Cipher Feed Back (CFB)
4. Output Feed Back (OFB)
5. Counter (CTR)

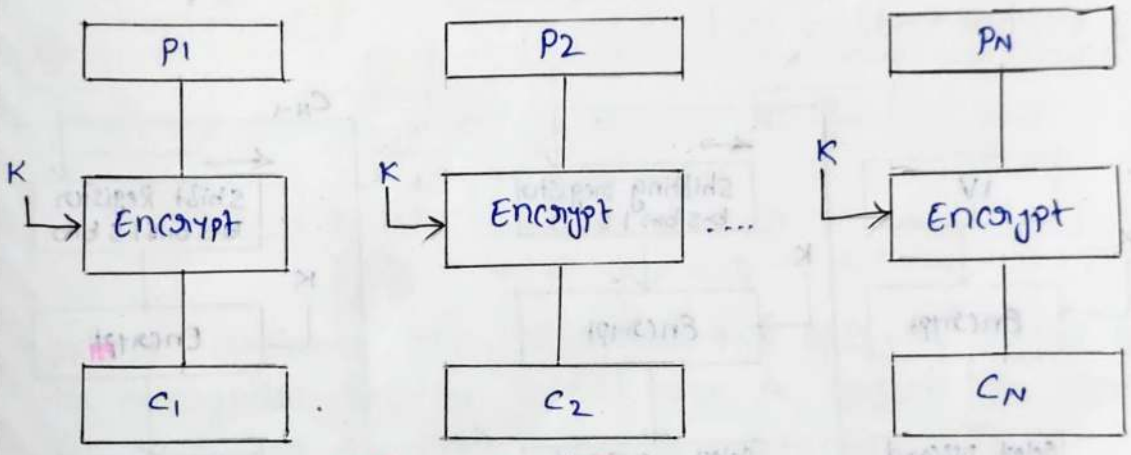
1. Electronic Code Book (ECB) :- mode :-

→ The simplest mode is ECB mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.

→ The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext.

→ ECB method is used in DES and AES.

→ For lengthy messages, the ECB mode may not be secure.



ECB Encryption

2. Cipher Block chaining (CBC) :- mode :-

→ To overcome the security deficiencies of ECB by using CBC. In CBC, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext

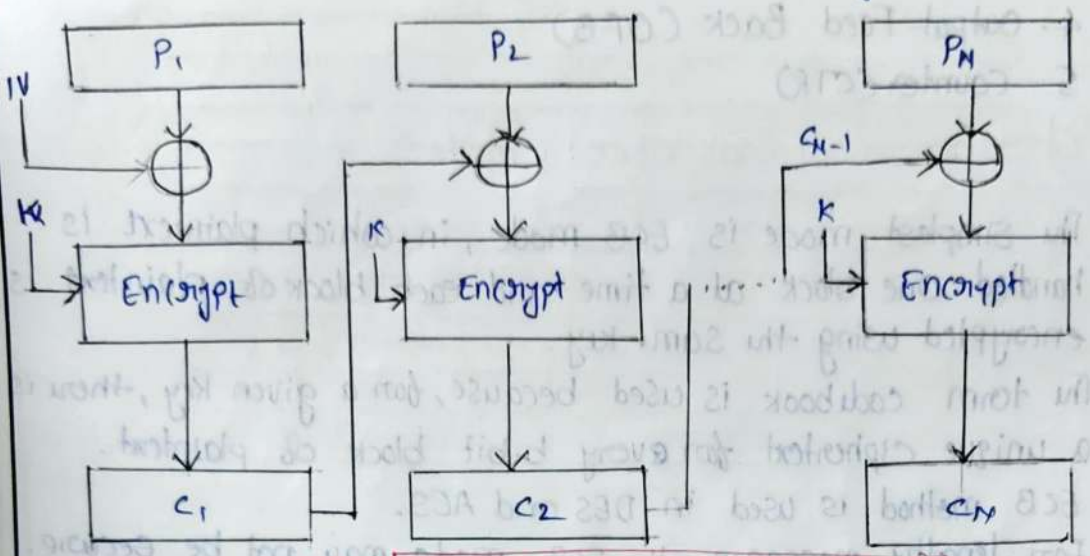
→ CBC mode is self Recovering

→ Encryption is not parallelized and decryption is parallelized with random access property.

block, the same key is used for each block. In effect, we have chained together the processing of the sequence of plain text blocks.

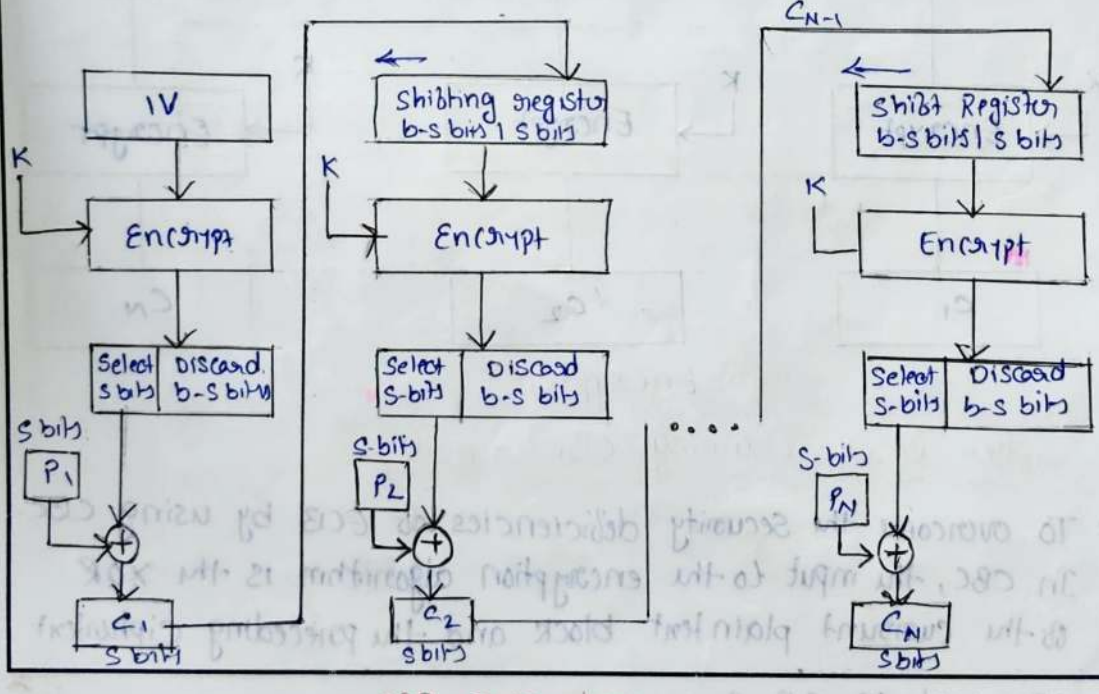
→ Because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than b bits.

→ In addition to its use to achieve confidentiality & Authentication



CBC [Encryption $(C_j = E(K, [C_{j-1} \oplus P_j])$]

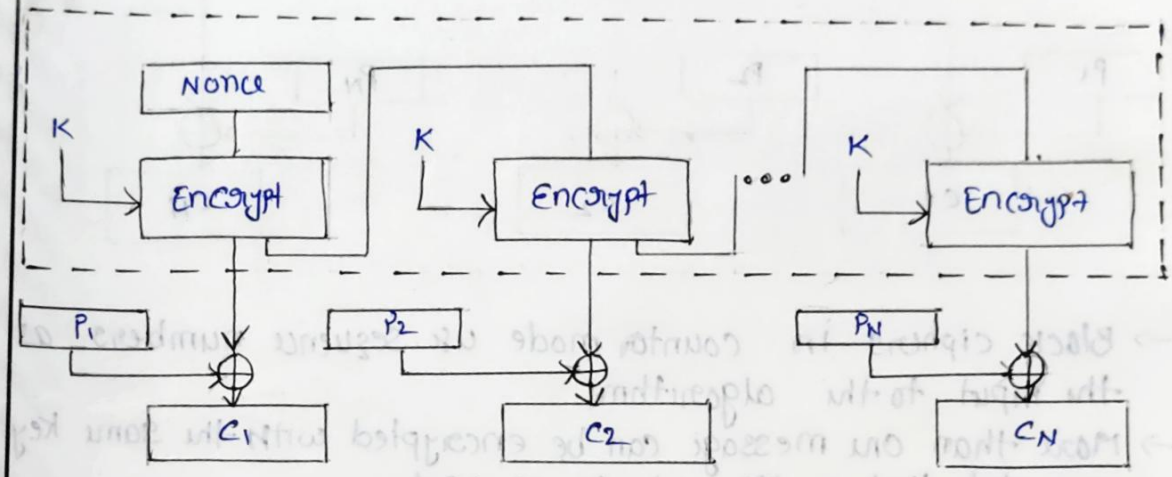
3. Cipher FeedBack Mode :- (CFB) :-



CFB Encryption

→ Data is encrypted in units that are smaller than a defined block size.
→ More than one message can be encrypted with the same key, provided that a different initialization vector is used.
→ CFB speed is the same as the block cipher.
→ Encryption is not parallelizable, decryption is parallelizable and has a random access property.
→ CFB is self Recovering with respect to synchronization errors as well
→ In order to guarantee that there is always some random looking ciphor text to apply to the actual plaint, the process is started with a block of random bits called the initialization vector (IV).

4. Output Feed Back (OFB) mode :-



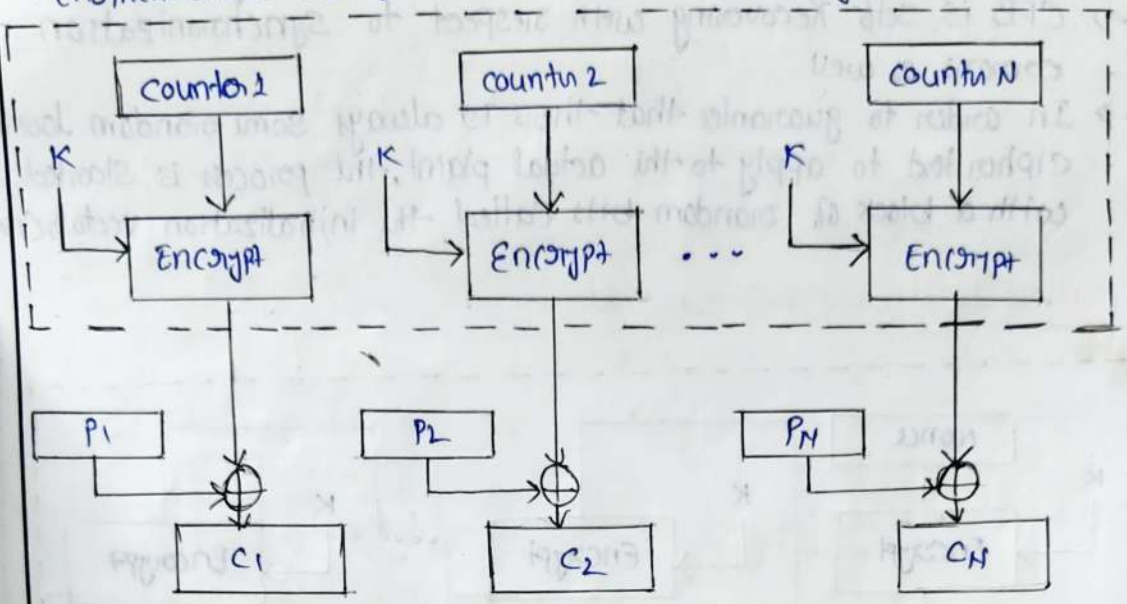
OFB Encryption.

→ OFB is similar to CFB structure. For OFB the output of the encryption function is fed back to become the input for encrypting the next block of plaintext.
→ In CFB the output of the XOR unit is fed back to become input for encrypting the next block.
→ The other difference is that the OFB mode operates on full blocks of plaintext and ciphertext, whereas CFB operates on an s-bit subset.

→ The advantage of OFB is that bit errors in transmission do not propagate.
→ The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

5. Counter mode

→ Counter mode has increased recently with application to ATM (Asynchronous transfer mode) network security and IPSec.



Counter mode Encryption

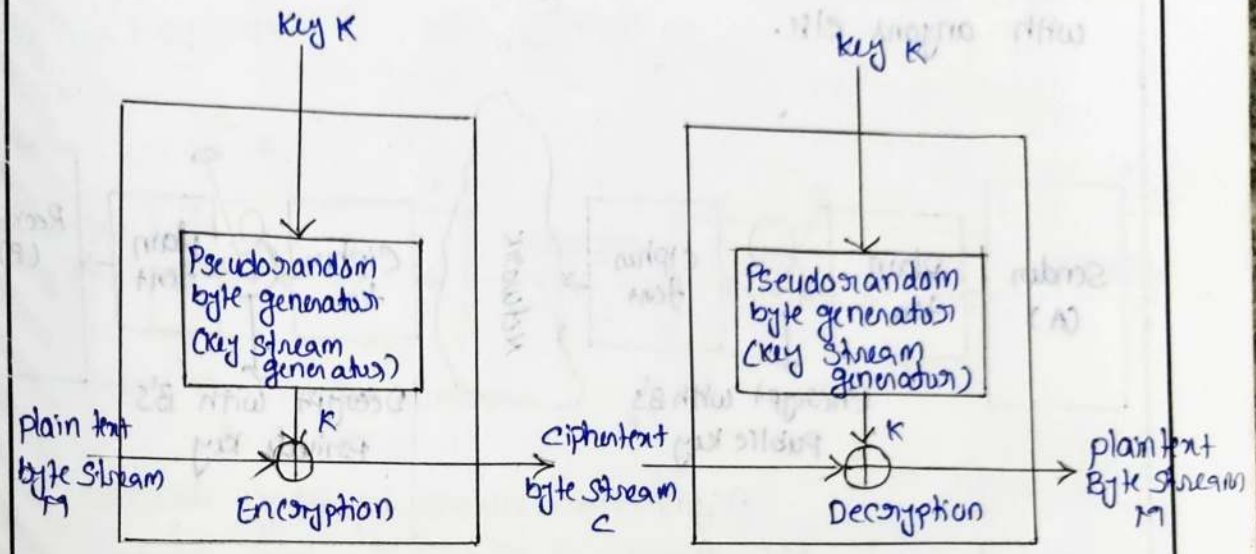
→ Block ciphers in counter mode use sequence numbers as the input to the algorithm.
→ More than one message can be encrypted with the same key provided that a different IV is used.
→ Plaintext is very easy to manipulate any change in ciphertext directly affects the plaintext.
→ Synchronization error is unrecoverable.
→ It is simple to implement
→ It provides confidentiality.
→ Random access of block is possible.
→ Efficiency is same as block cipher
→ It is useful for high speed requirements.

Stream ciphers :-

→ A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time.

→ In stream cipher a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random.

→ The output of the generator called a keystream is combined one byte at a time with the plaintext stream using the XOR operation.



Stream Cipher Diagram

Advantages :-

→ Speed of transformation

→ Low error propagation

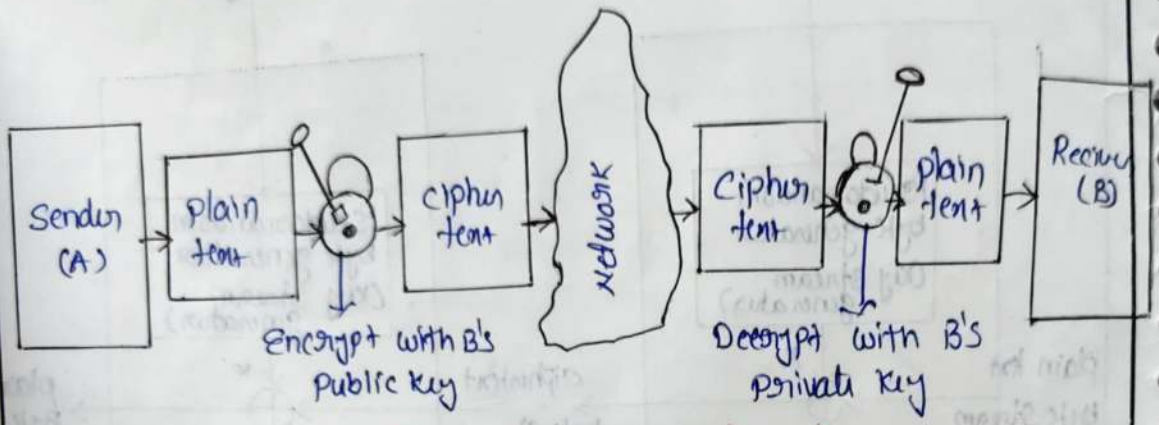
Disadvantage :-

→ Low diffusion

→ Susceptibility to malicious insertation and modifications.

Asymmetric-key cryptography:

→ It is also called public key cryptography, two different keys are used. One key for encryption and only the other corresponding key must be used for decryption.
→ No other key can decrypt the message - not even the original key used for encryption.
→ The beauty of this scheme is that every communicating party needs just a key pair for communicating with any number of other communicating parties.
→ Once someone obtains a key-pair he/she can communicate with anyone else.



Asymmetric-key cryptography

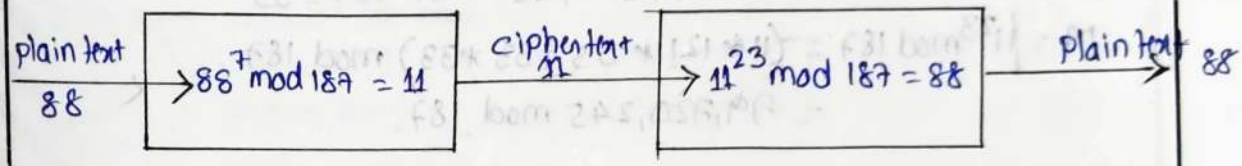
RSA Algorithm:

→ One of the first sucessful responses to the challenge (public key cryptography) was developed in 1977 by Ron Rivest, Adi Shamir and Len Adleman at MIT
→ The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
→ The RSA scheme is a cipher in which the plaintext and ciphertext are integers b/w 0 and n-1 for some n. A typical size for n is 1024 bits or 309 decimal digits. i.e $n < 2^{1024}$.

Algorithm :

1. Select P, Q two prime numbers, $P \neq Q$ (Private, choosen)
2. Calculate $n = P * Q$ (Public, calculate)
3. Calculate $\phi(n) = (P-1)*(Q-1)$
4. Select integer e. $\gcd(\phi(n), e) = 1$; (Public, choosen).
   $1 < e < \phi(n)$
5. Calculate d $d = e^{-1} \pmod{\phi(n)}$ (Private, calculated). (Euler's rule)
6. Public key $PU = \{e, n\}$
7. Private key $PR = \{d, n\}$
8. ciphertext $C = M^e \bmod n$ $(M < n)$
9. plain text $M = C^d \bmod n$

Example :



1. Select two prime numbers
   $P = 17$, $Q = 11$
2. Calculate $n = PQ = 17 * 11 = 187$
3. Calculate $\phi(n) = (P-1) * (Q-1) = 16 * 10 = 160$
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$ we choose $e = 7$.
5. determine d such that $de = 1 \pmod{160}$ and $d < 160$. the correct value is $d = 23$. because $23 * 7 = 161 = (1 * 160) + 1$; d can be calculated using extended Euclid's Algorithm.
6. Public key $PU = \{7, 187\}$
7. Private key $PR = \{23, 187\}$
8. $M = 88$ then encryption
9. $C = 88^7 \bmod 187$.

$C = 88^7 \bmod 187 = [88^1 \bmod 187 * 88^2 \bmod 187 * 88^4 \bmod 187] \bmod 187$

$88^1 \bmod 187 = 88$

$88^2 \bmod 187 = 7744 \bmod 187 = 77$

$88^4 \bmod 187 = 59,969,536 \bmod 187 = 132$

$C = 88^7 \bmod 187 = (88 * 77 * 132) \bmod 187 = 11$

$\therefore 894,432 \bmod 187 = 11$

10. decryption.

$M = 11^{23} \bmod 187$

$11^{23} \bmod 187 = (11^1 \bmod 187 * 11^2 \bmod 187 * 11^4 \bmod 187 * 11^8 \bmod 187 * 11^8 \bmod 187) \bmod 187$

$11^1 \bmod 187 = 11.$

$11^2 \bmod 187 = 121$

$11^4 \bmod 187 = 14,641 \bmod 187 = 55,$

$11^8 \bmod 187 = 214,358,881 \bmod 187 = 33$

$M = 11^{23} \bmod 187 = (11 * 121 * 55 * 33 * 33) \bmod 187.$

$= 79,720,245 \bmod 187.$

Example 2 :-

1. Select two prime numbers

$p = 47, q = 17$

2. Calculate $n = p * q = 47 * 47 = 119.$

3. $\phi(n) = 6 * 16 = 96$

4. $\gcd(\phi(n), e) = 1$

$\gcd(96, e) = 1 \quad \therefore e = 5$

5. $d * e \bmod \phi(n) = 1$

$d * 5 \bmod 96 = 1$

$d = 77$

6. $C = M^e \bmod n$

$C = 10^5 \bmod 119 = 40$

7. $M = C^d \bmod n$

$= 40^{77} \bmod 119$

$M = 10.$

Example:- Using RSA algorithm calculate the following

i. Given n=221 and e=5 find d?

$n = 221$

$n = P * Q$

$221 = P * Q$

$221 = 13 * 17$

So. $P = 13, Q = 17$

$\phi(n) = (P-1)(Q-1) = (17-1) * (13-1) = 192$

$e * d = 1 \bmod \phi(n)$

$5 * d = 1 \bmod 192$

$d = 77.$

ii. Given $n = 3937$ and $e = 17$.

$n = 3937$

$n = P * Q$

$3937 = P * Q$

$3937 = 127 * 31$

So $P = 127, Q = 31$

$\phi(n) = (P-1)(Q-1) = (127-1)(31-1) = 3780$

$e * d = 1 \bmod \phi(n)$

$17 * d = 1 \bmod (3780)$

$d = 223.$

//Elgamal Key Generation:

Attacks on RSA:-

→ Attacks on RSA algorithms are as follows:

1. Brute force:-

This involves trying all possible private keys.

2. Mathematical attacks :-

→ This involves the factoring the product of two primes

3. Timing Attacks :

→ These depends on the running time of the description algorithm.

4. Choosen Ciphertext Attacks :

→ This type of attack exploits properties of the RSA algorithm.

Note:

→ RSA can be used both for encryption as well as for digital signatures.

ElGamal Cryptography :-

→ It is an public key cryptosystem. It uses asymmetric key encryption for communicating between two parties and encrypting the message.

→ There are three aspects are involved.

1. ElGamal key generation
2. ElGamal key Encryption
3. ElGamal key Decryption.

1. ElGamal Key Generation :-

This involves the following steps :

1. Select a large prime number called P. This is the first part of the encryption key or public key.
2. Select the decryption key or private key D.
3. Select the second part of the encryption key or public key E_1.
4. find the third part of the encryption key or public key E_2

$$E_2 = E_1^D \bmod P$$

5. The public key is (E_1, E_2, P) and private key is D.

Example :-

$P = 11, E1 = 2, D = 3.$

Then $E_2 = E_1^D \bmod P$

$E_2 = 2^3 \bmod 11 = 8.$

then public key $(2, 8, 11)$, private key is 3.

2 ElGamal Key Encryption :-

→ This involves the following steps:

1. Select a random integer R.
2. compute the first part of the ciphertext $C_1 = E_1^R \bmod P.$
3. compute the second part of the ciphertext $C_2 = (PT \times E_2^R) \bmod P$
4. The final cipher text is (C_1, C_2).

Example :

Let $R = 4$, plain text $PT = 7.$

$C_1 = E_1^R \bmod P = 2^4 \bmod 11 = 5$

$C_2 = (PT \times E_2^R) \bmod P = (7 \times 2^8) \bmod 11 = (7 \times 4096) \bmod 11 = 6.$

cipher text is $(5, 6)$.

3. ElGamal Key Decryption :-

→ This involves the following step :

compute the plain text PT using the formula

$$PT = [C_2 \times (C_1^D)^{-1}] \bmod P$$

Example

$PT = [6 \times (5^3)^{-1}] \bmod 11$

$= [6 \times 3] \bmod 11$

$PT = 7$, which was our original plain text

Diffie-Hellman :-

- The Diffie-Hellman key agreement protocol was developed by Diffie and Hellman in 1976.
- The purpose of the algorithm is to enable two users to securely exchange a key that can be used subsequent symmetric encryption of messages. The algorithm itself is limited to the exchange of secret values.

Algorithm :-

1. Select two numbers
   a. prime number q
   b. α an integer that is a primitive root of q.

→ Suppose the users A and B wish to exchange a key.

1. User A select a random integer $X_A < q$ and computes

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

2. User B select a random integer $X_B < q$ and computes

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

3. Both side keeps the X value private and makes the Y value available publicly to the other side.
4. User A computes the key as

$$K = (Y_B)^{X_A} \bmod q$$

5. User B computes the key as

$$K = (Y_A)^{X_B} \bmod q$$

$$K = (Y_B)^{X_A} \bmod q$$
$$= (\alpha^{X_B} \bmod q)^{X_A} \bmod q$$
$$= (\alpha^{X_B})^{X_A} \bmod q$$

$= \alpha^{X_B X_A} \bmod q$

$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$

$K = (Y_A)^{X_B} \bmod q$

Example:

$q = 353$

$\alpha = 3$

$X_A = 97$

$X_B = 223$

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

$= (3)^{97} \bmod 353$

$Y_A = (1.9080 \times 10^{93}) \bmod 358 = 40$

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

$= (3)^{233} \bmod 353$

$= (1.4765 \times 10^{111}) \bmod 353$

$Y_B = 248$

$K = (Y_B)^{X_A} \bmod q = (248)^{97} \bmod 353 = 160$

$K = (Y_A)^{X_B} \bmod q = (40)^{233} \bmod 353 = 160$

Example:

$q = 71$

$\alpha = 7$

$X_A = 5$

$X_B = 12$

$Y_A = \alpha^{X_A} \bmod q = 7^5 \bmod 71 = 51$

$Y_B = \alpha^{X_B} \bmod q = 7^{12} \bmod 71 = 4$

$K = (Y_B)^{X_A} \bmod q$

$K = (4)^5 \bmod 71 = 30$

Note:- Does not protect against Man-in-the-middle Attacks

Knapsack Algorithm :

→ Actually in 1978, Ralph Merkle and Martin Hellman developed the first algorithm for public-key encryption, called as knapsack algorithm.

→ It is based on knapsack problem.

→ This is actually a simple problem. Given a pile of items, each with different weights, is it possible to put some of them in a bag or knapsack in such a way that the knapsack has a certain weight ?

→ i.e. If $m_1, m_2, \ldots m_n$ are the given values and S is the sum, find out bi so that :

$$S = b_1m_1 + b_2m_2 + \ldots + b_nm_n.$$

→ Each bi can be 0 or 1. A 1 indicates that the item is in the knapsack and a 0 indicates that it is not.

→ A block of plaintext equal in length to the number of items in the pile would select the items in the knapsack. The ciphertext is the resulting sum.

→ for example, if knasack is 1, 7, 8, 12, 14, 20 then the plain text and resulting plain text is.

plain Text	0 1 1 0 1 1	1 1 1 0 0 0	0 1 0 1 1 0
Knapsack	1 7 8 12 14 20	1 7 8 12 14 20	1 7 8 12 14 20
Ciphertext	7+8+14+20=49	1+7+8=16	7+12+14=33.

Knapsack Example.