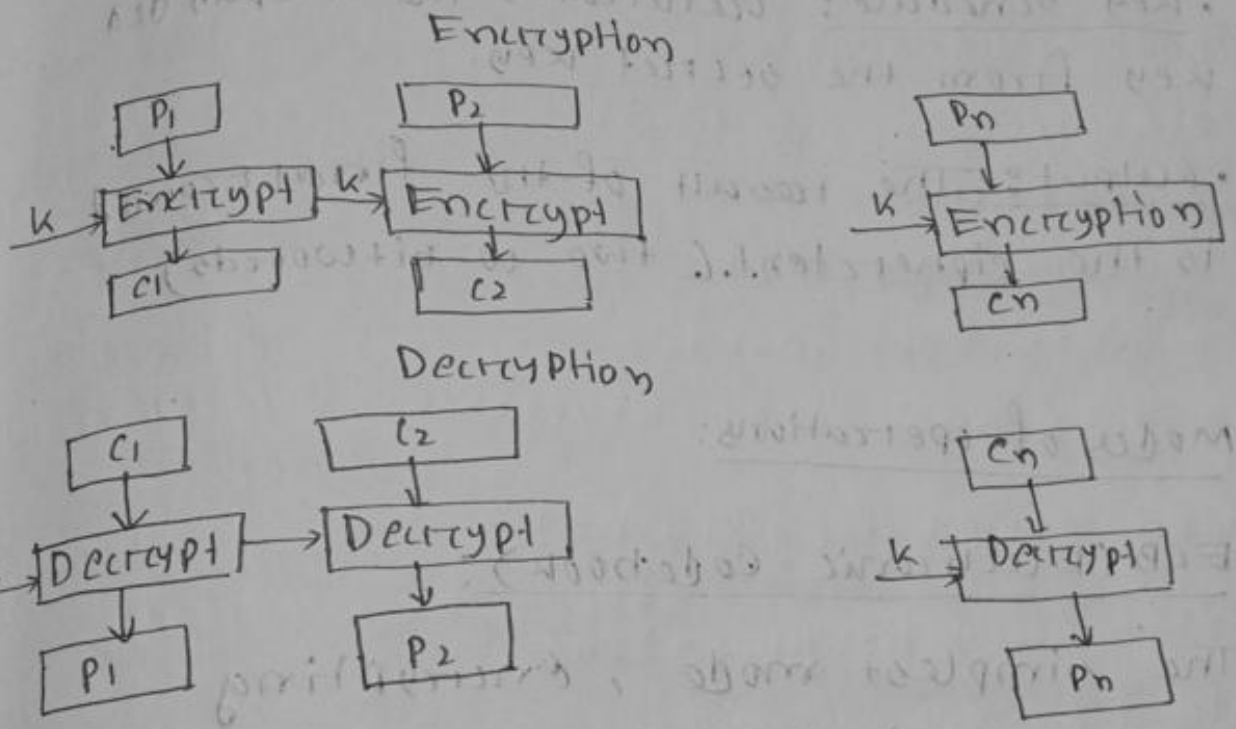
RC5: is a symmetric block cipher known for its simplicity and flexibility in key and block sizes.



RC5 Block Diagram:

- Input: The plaintext is divided into two words (w bits each)
- Initial Key Addition: The input words a

Combined with a portion of the expande

key

- Mixing (circular-shifts): These operations (XOR and Rotations) are applied iteratively in multiple rounds.
- Key schedule: Generates the expanded key from the secret key.
- Output: The result of the final round is the ciphertext (two w-bit words).

Modes of operations:

i) ECB (Electronic Codebook):

The simplest mode, encrypting each block independently, suitable for short, random data but can reveal patterns in non-random.

ii) CBC (Cipher Block chaining):

Each block is XORed with the previous ciphertext block before encryption. providing better security than ECB

iii) OFB (Output Feedback):

Uses the previous ciphertext block to generate the keystream, making it suitable for noisy channel.

iv) CFB (Cipher feedback):

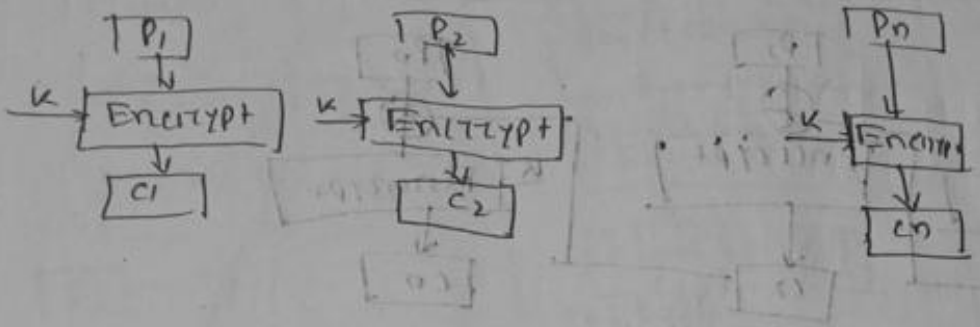
converts the block cipher into a stream cipher by feedingback previous ciphertext blocks.

v) Counter (CTR):

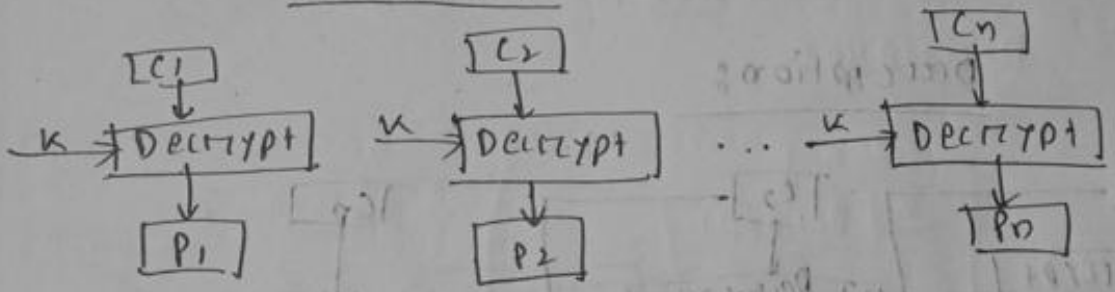
Uses a counter to generate the keystream allowing for parallel encryption and decryption.

ECB

Encryption



Decryption



Advantages of ECB:

→ Parallel encryption of blocks of bits is possible thus it is a faster way of encryption.

→ simple way of the block ciphers.

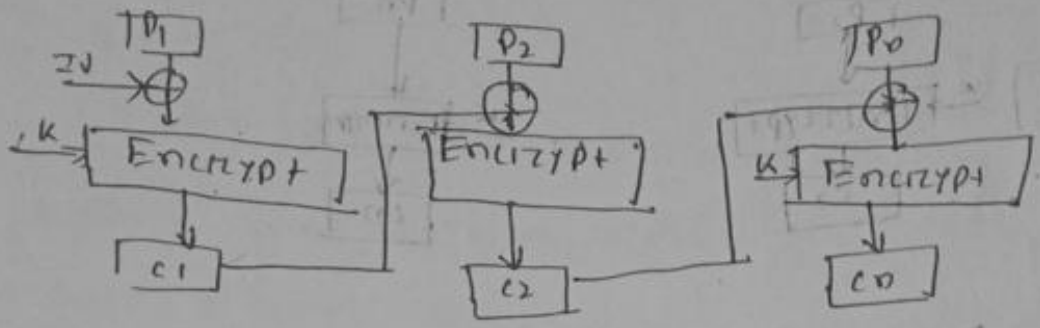
Disadvantages of ECB:

→ Prone to encryption of blocks of bits is possible

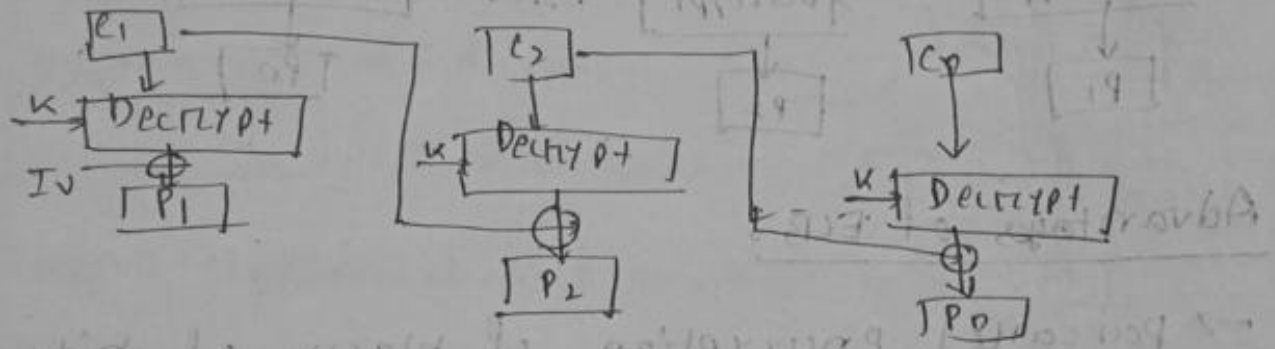
→ simple way of the block ciphers

CBC

Encryption



Decryption:



advantages:

→ CBC works well for input streams that b bits.

→ CBC is a good authentication mechanism

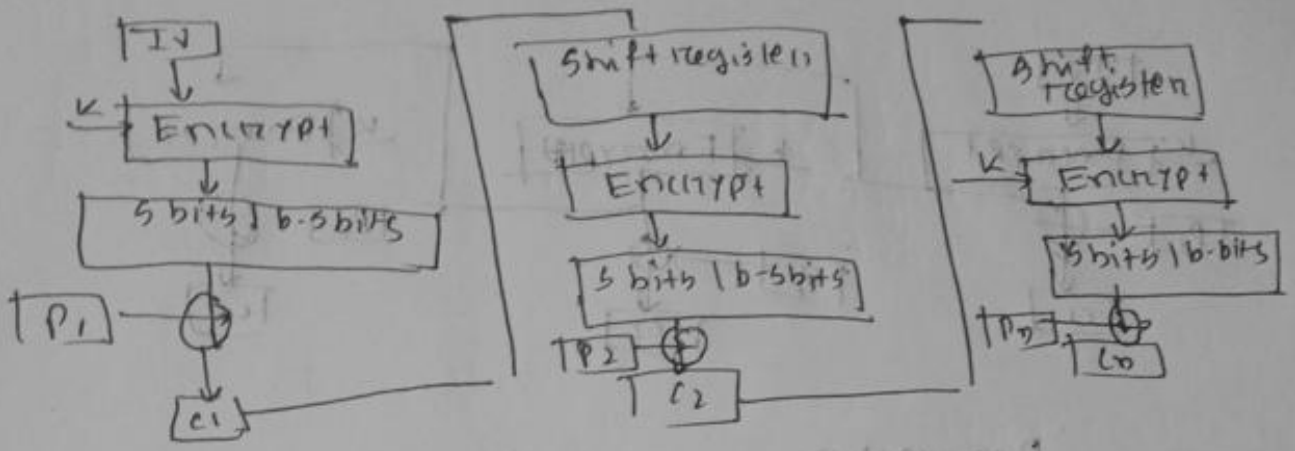
→ Better resistive nature towards encryptanalysis than ECB.

Disadvantages:

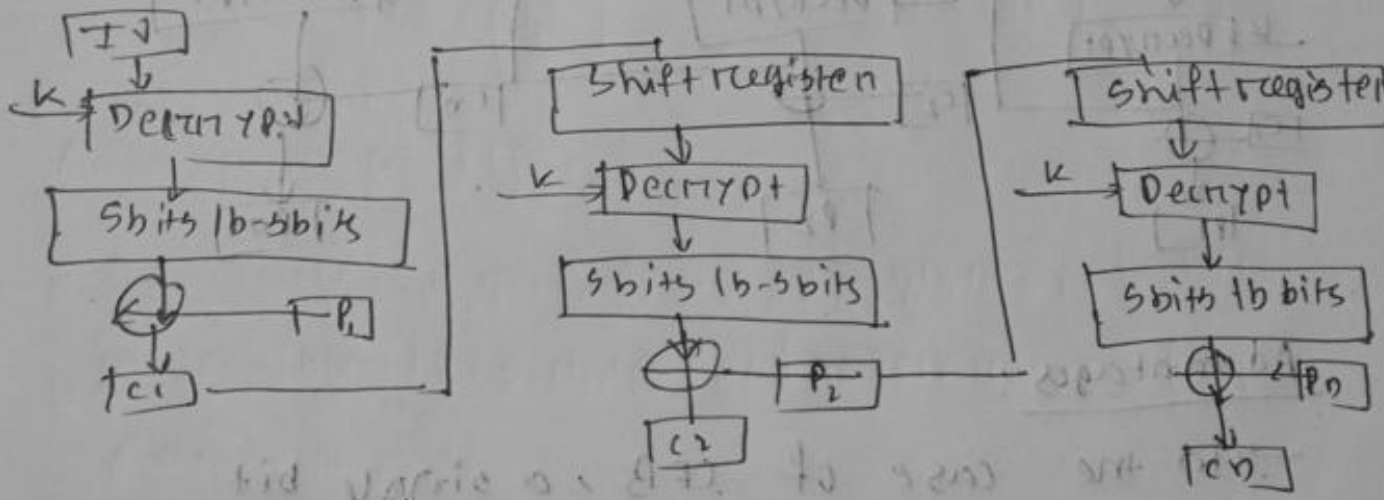
→ Requires the previous ciphertext block for encryption and decryption.

CFB

Encryption



Decryption



advantages of CFB :

→ there is some data loss due to the use of shift registers, thus it is difficult for applying cryptanalysis

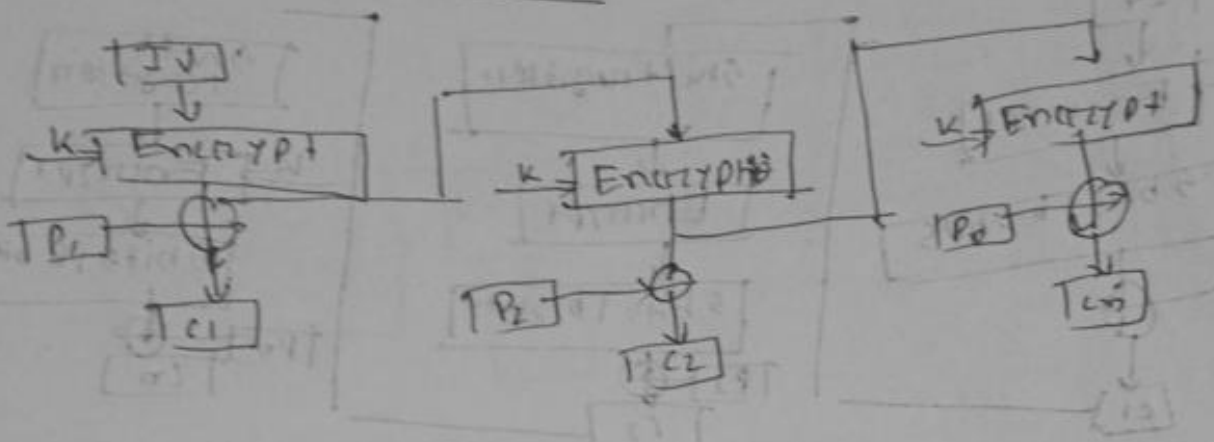
→ can handle data ~~streams~~ of any size.

Disadvantages:

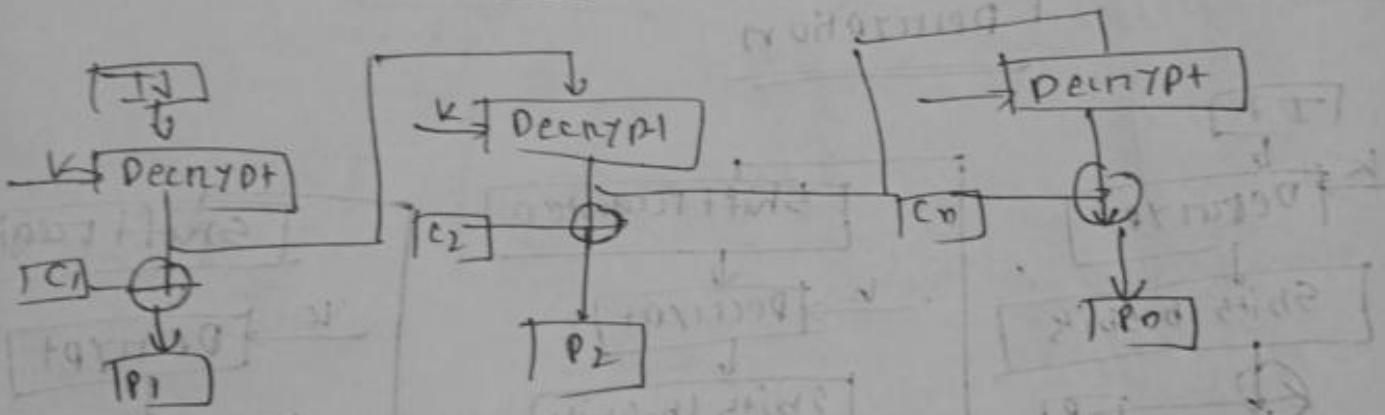
→ slightly more complex and can propagate errors

OFB

Encryption



Decryption



Advantages:

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

Disadvantages:

→ It is more susceptible to a message stream modification attack than CFB

Java implementation of RC5:

```
import java.nio.ByteBuffer;
import java.util.Arrays;

public class RC5 {
private final int w= 32;
private final int r= 12;
private final int b = 16;
private final int[] s;
private final int pw = 0xb7e15163, qw= 0x9e3779b9;

public RC5 (byte[] key) {
int c = (key.length +3)/4;
int[] L = new int[c];

for (int i=0; i< key.length; i++) {
L[i/4] = ((key[i] & 0xFF) << (8 * (i%4)));}


int t = 2*(r+1);
s = new int[t];
s[0] = pw;
for(int i=1; i<t; i++) {
   s[i] = s[i-1]+qw; }
```

```
int A=0; B=0, i=0, j=0;
int v = 3 * Math.max(c,t);
for(int j=0; j<v; j++){
A = S[i] = Integer.rotateLeft((S[i]+A+B),3);
B = L[j] = Integer.rotateLeft((L[i]+A+B),(A+B));
i = (i+1)%t;
j = (j+1)%c;
}
}

public int[] encrypt(int[] pt){
int A = pt[0]+S[0];
int B = pt[1]+S[1];
for (int i=1; i<=r; i++){
A= Integer.rotateLeft(A^B,B)+S[2*i];
B = Integer.rotateLeft(B^A,A)+S[2*i+1];

return new int[]{A,B}; }

public int[] decrypt(int[] ct){
int A = ct[0];
int B = ct[1];
for(i=r; i>=1; i--){
```

```
B = Integer.RotateRight(B-S[2*i+1], A)^A;
A= Integer.rotateRight(A-S[2*i],B)^B; }
A -= S[0];
B -= S[1];
return new int[]{A,B}; }
Public static void main(String[] args) {
byte[] key = "MySecretKey12345".getBytes();
RC5 rc5 = new RC5(key);
int[] plaintext={0x12345678, 0x9ABCDEF0};
System.out.printf("Plaintext: %08x %08x\n",
                  Plaintext[0], Plaintent[1]);
Int[] Ciphertext = rc5.encrypt(Plaintent);
System.out.printF(" Encrypted: %008x %0.8x
                  Ciphertext[0], Ciphertext[1]);
int[] decrypted = rc5.decrypt(Ciphertext);
System.out.printf("Decrypted : %08x %008x\n
                  decrypted[0], decrypted[1]); }}
```

Sample output:

Plaintent: 12345678 9ABCDEF0
Encrypted: E2C3456 5E91A7B9
Decrypted: 12345678 9ABCDEF0