Assignment-6

Shafinaj Jerin Asha
IT-21031

Name of the Assignment:

Modes of Operation and RC5 - Block Diagram and Java Implementation and output.

Modes of Operation:

1. ECB - Electronic Codebook
2. CBC - Cipher Block Chaining
3. CFB - Cipher Feedback
4. OFB - Output Feedback
5. CTR - Counter

Description:

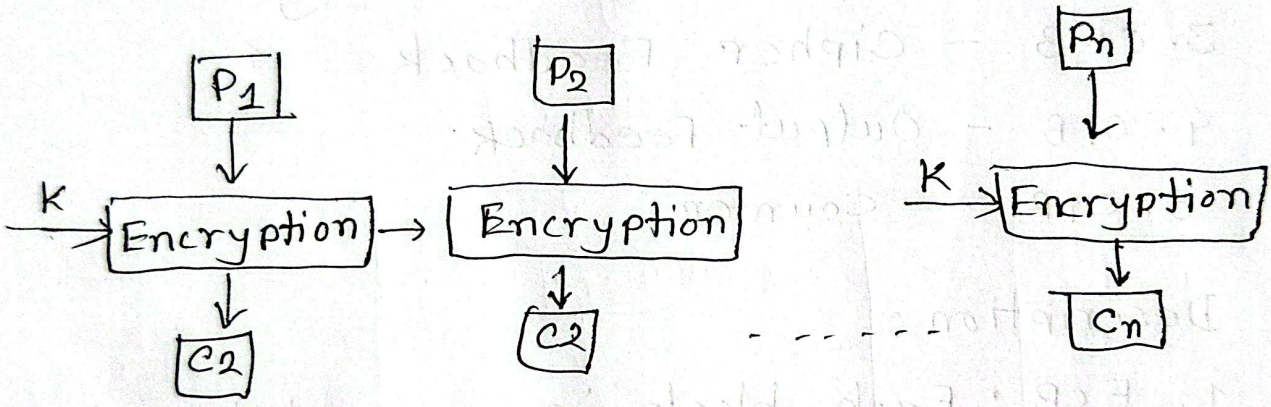
1. ECB: Each block is encrypted independently. Not secure for patterns.
2. CBC: XORs each plaintext block with previous ciphertext block before encrytion.
3. Converts block cipher into a self-synchronizing stream cipher.

4. OFB : Turns block cipher into a synchronous stream cipher.

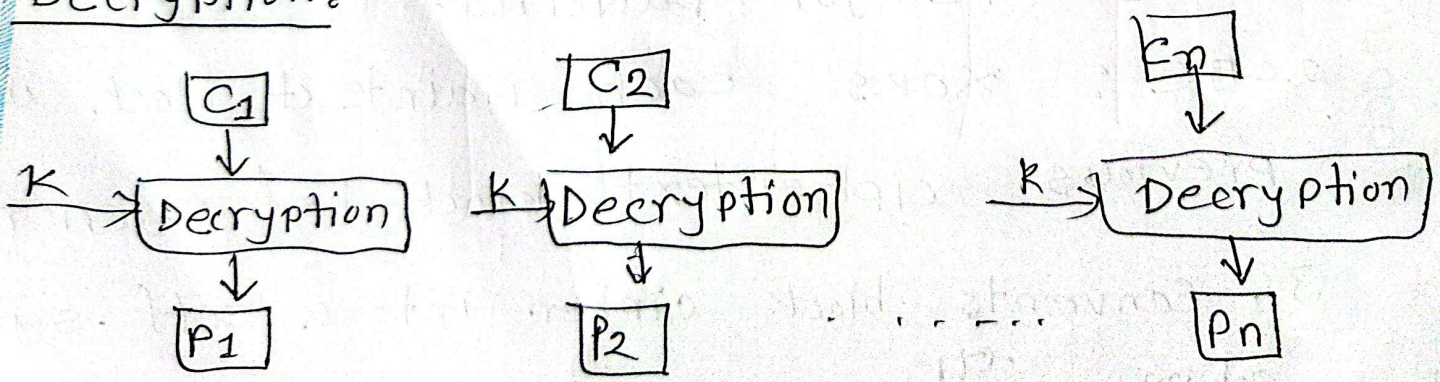
5. CTR : Uses a counter that gets encrypted and XORed with plaintext Fast and parallelizable.

The procedure of ECB is illustrated below:

Encryption :



Decryption :

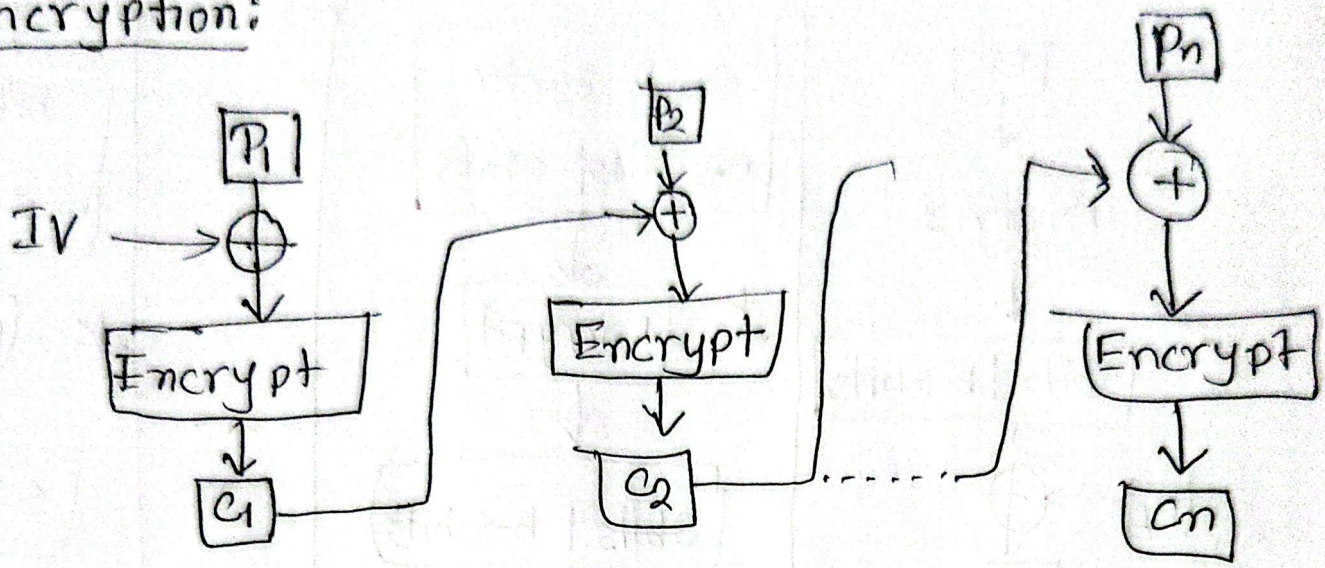


TOPIC NAME IT-21031 DAY

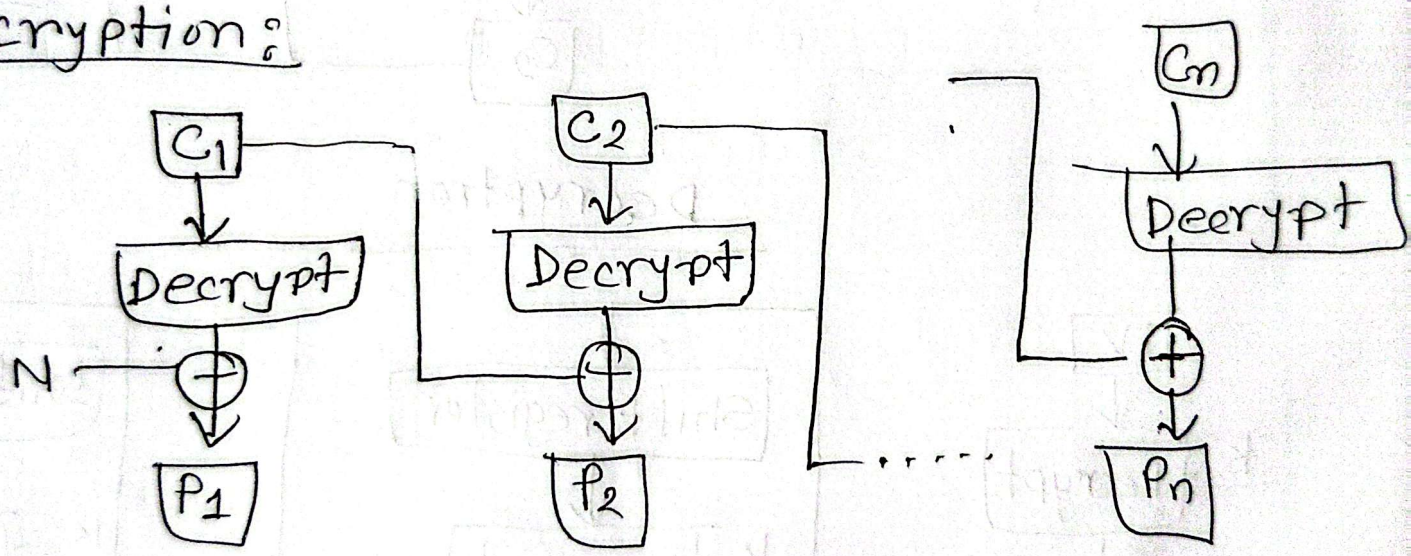
TIME DATE / /

CBC Block Diagram:

Encryption:



Decryption:

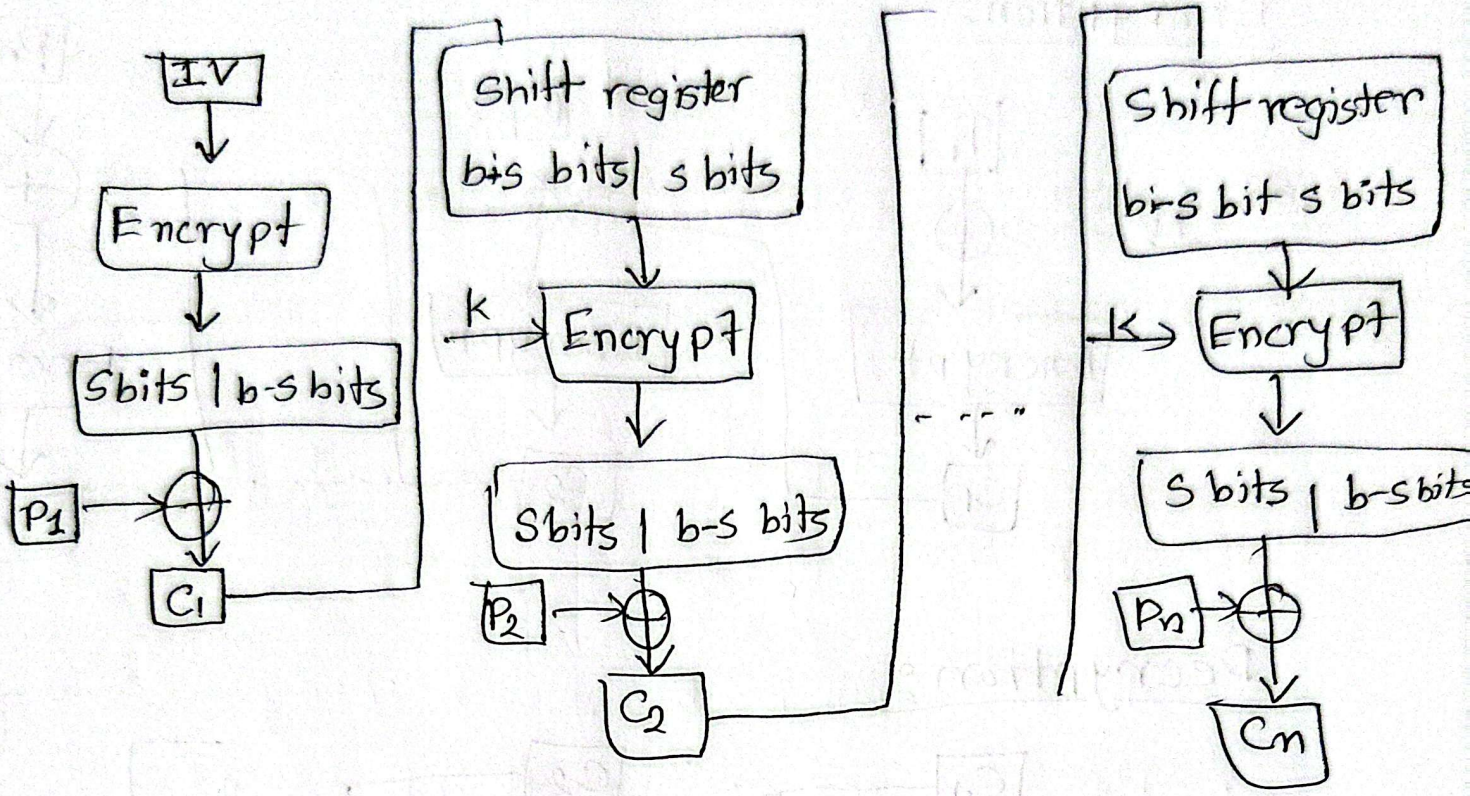


CFB Block Diagram:

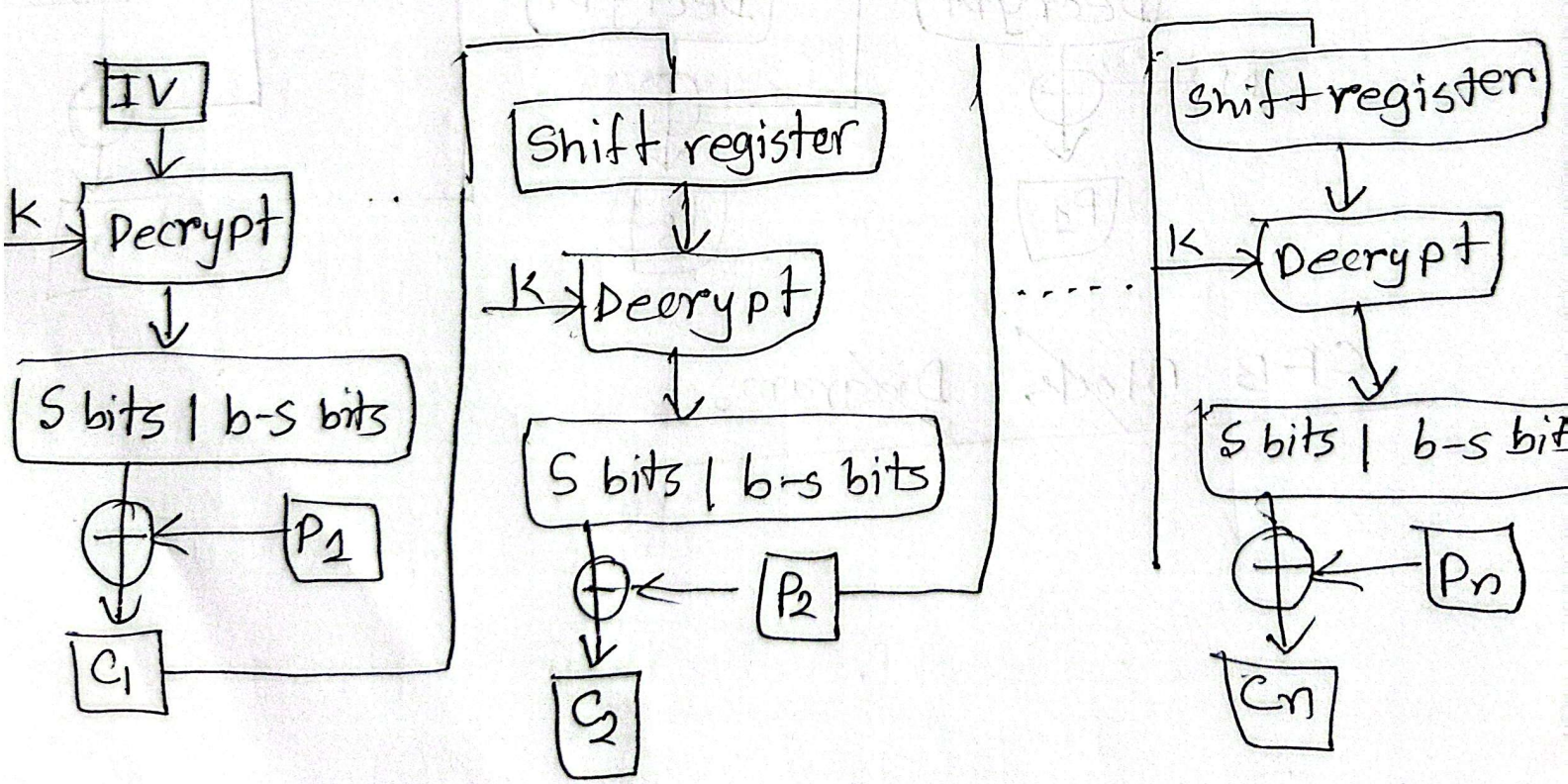
TOPIC NAME : IT-21031 DAY : TIME : DATE : / /

CFB Block Diagram :

Encryption



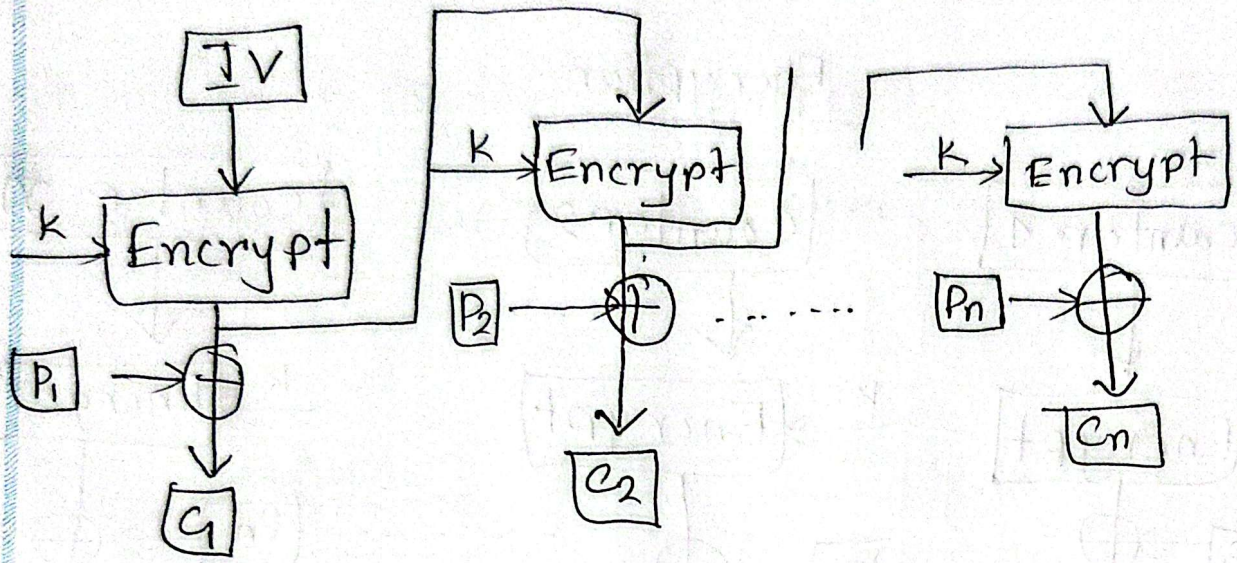
Decryption



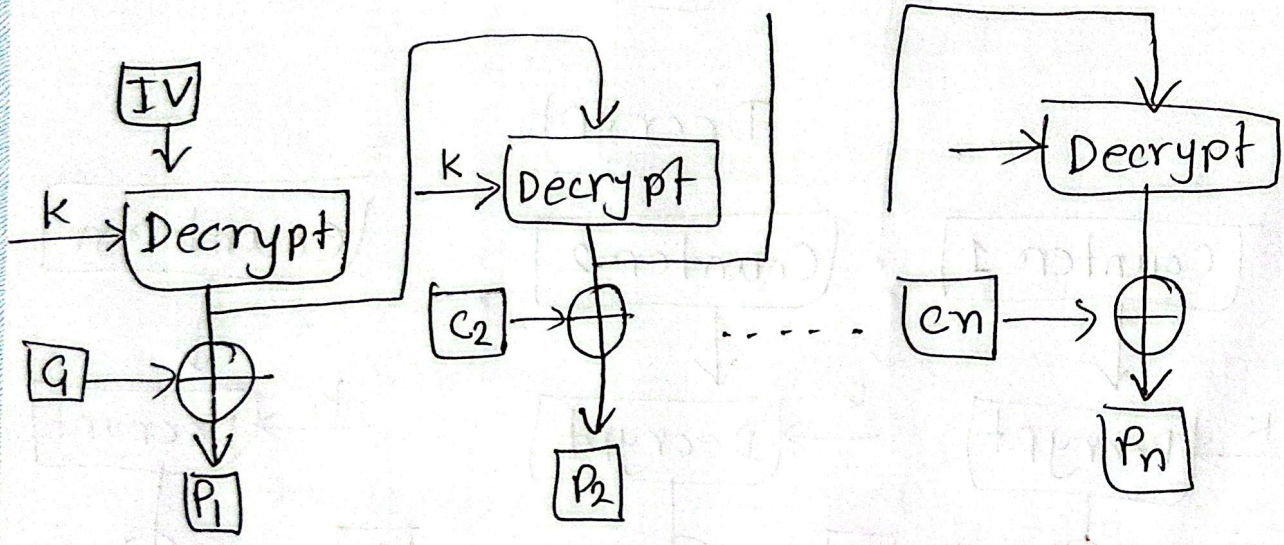
TOPIC NAME : IT-21031 DAY :

TIME : DATE : / /

OFB Block Diagram :



Dycryption

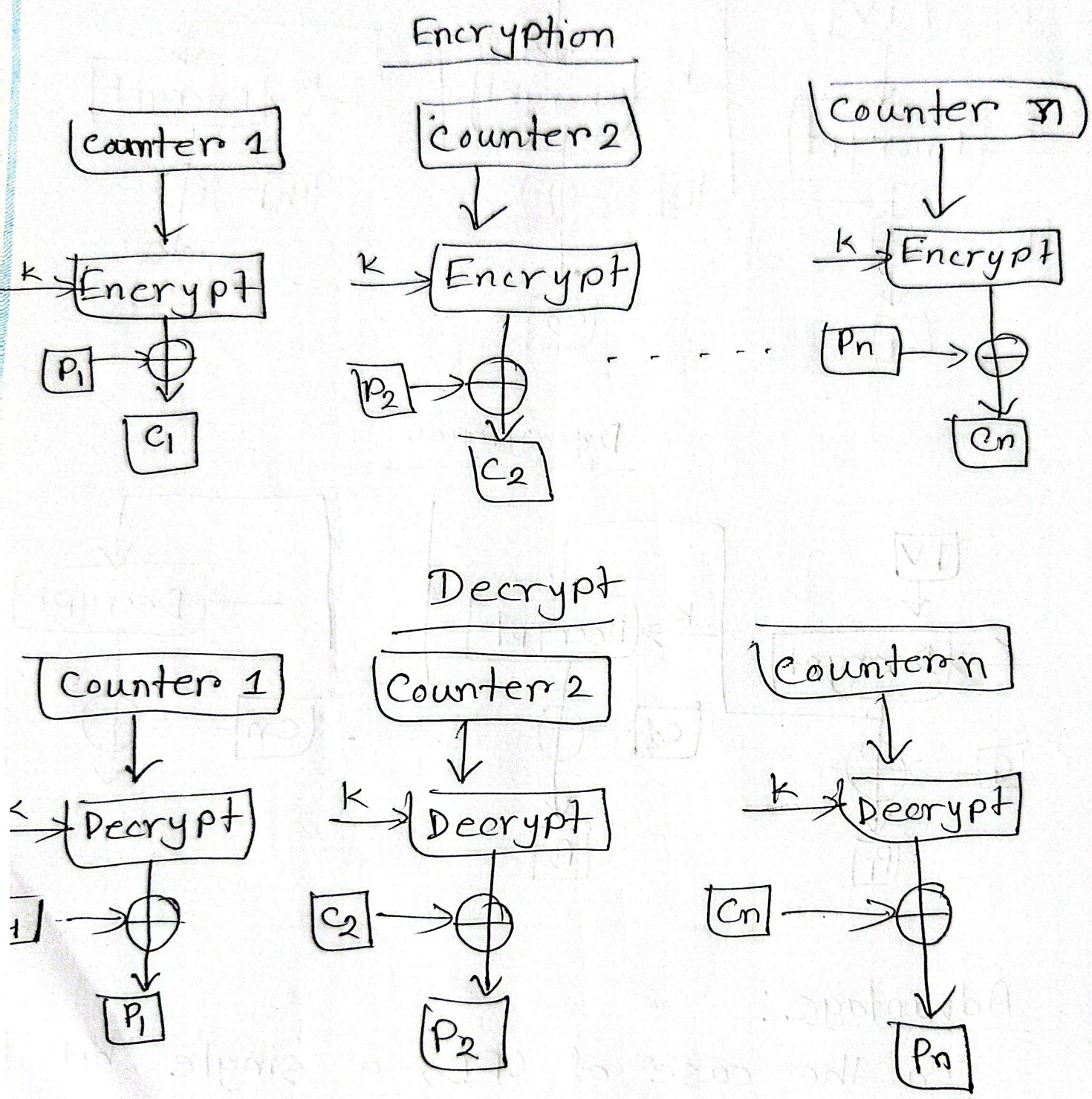


Advantage :

In the case of CFB, a single bit error in a block is propagated to all subsequent blocks.

TOPIC NAME : IT-21031 DAY : TIME : DATE : / /

Block Diagram of CTR:



TOPIC NAME : IT-21031 DAY

TIME : DATE : / /

Introduction of RC5:

RC5 is a fast, simple, and secure symmetric key block cipher designed by Ron Rivest in 1994.

key features:

- Parameterizable:
  - * Word size (32 bits)
  - * Number of round (12)
  - * key length (8 bytes)
- Uses:
  - * Bitwise operations : XOR, shift, rota
  - * Modular addition

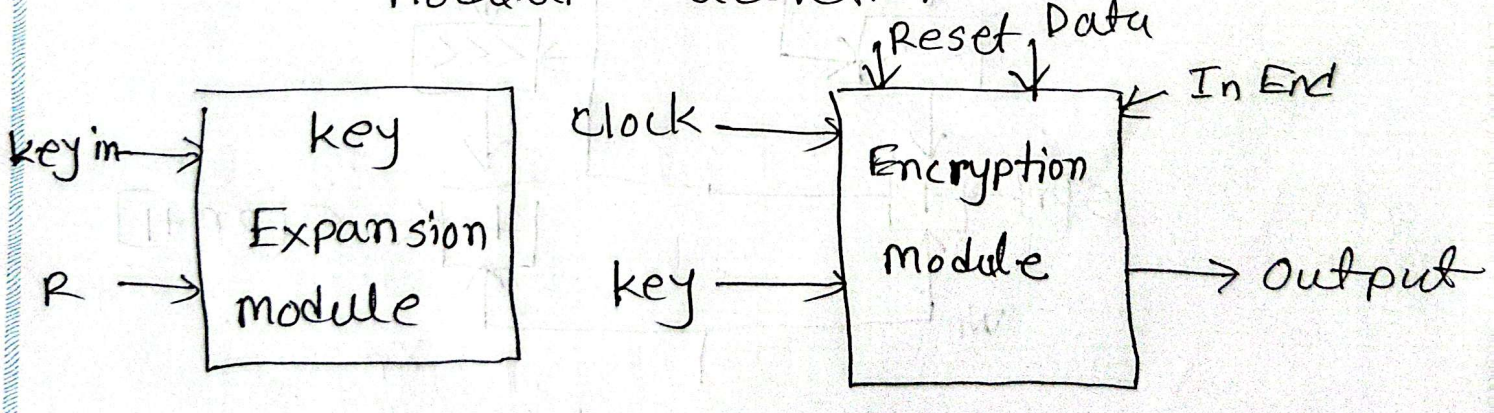
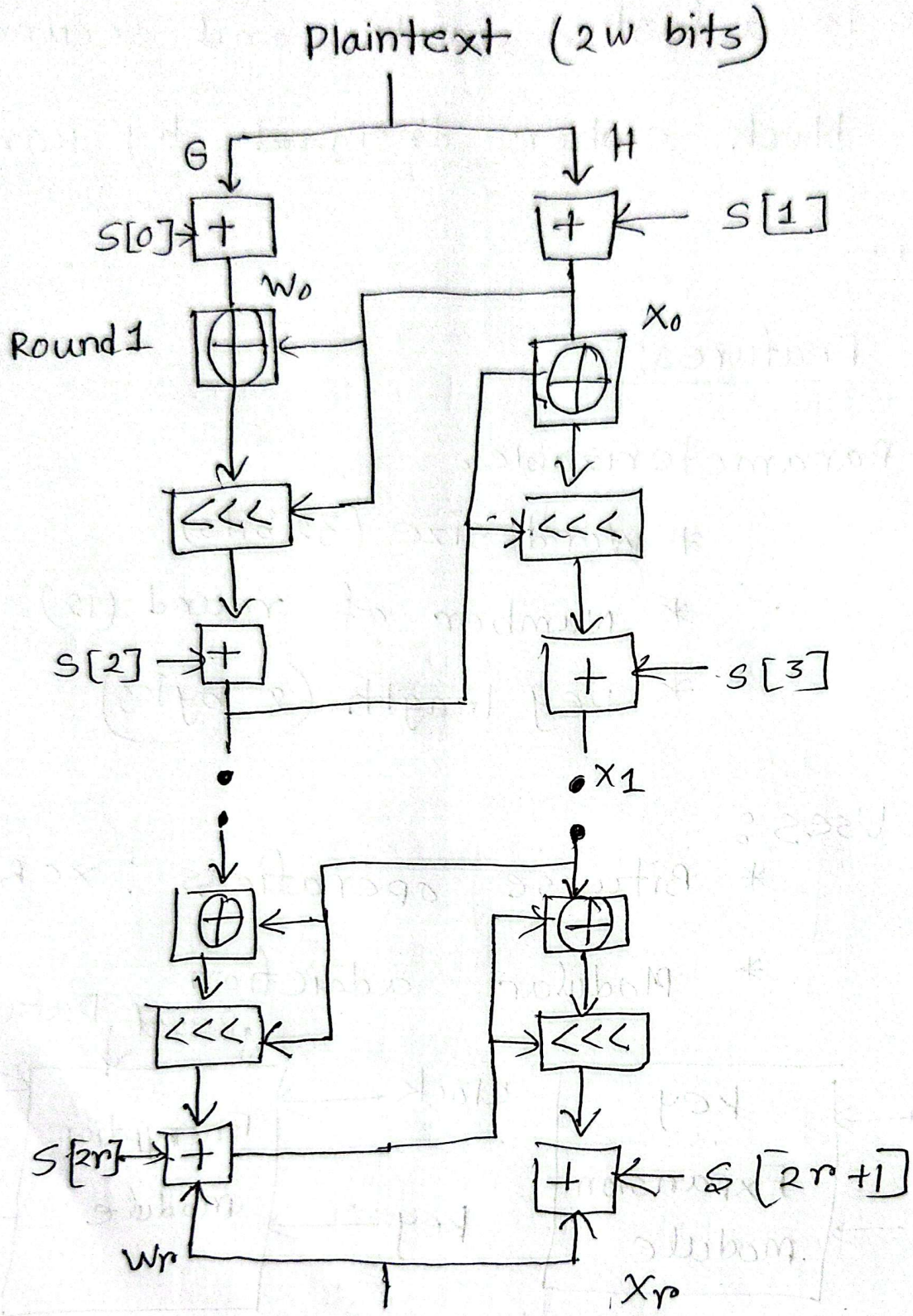


Fig: RC5 Encryption Block Diagram

TOPIC NAME : JT-21031 DAY :

TIME DATE : / /

RC5 Block Diagram:



TOPIC NAME: IT-21031 DAY
TIME: DATE: / /

Java Implementation:

```
Public class RC5 {
    private static final int WORDSIZE = 32;
    private static final int R = 12;
    private static final int B = 8;
    private static final int C = B/4;
    private static final int T = 2*(R+1);
    private int [] S = new int [T]
    private static final int P = 0xB7E15163;
    private static final int Q = 0x9E3779B9;
    public RC5 (byte [] key) {
        key schedule (key);
    }
private void keyschedule (byte [] key) {
    int [] L = newint [c];
    for (in i=0; i<B; i++) {
    L[i/4] = (L[i/4]< B) +(key[i] & 0xFF);
    }
```

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

public int [] decrypt (int [] ct) {
    int B = ct [1];
    int A = ct [0];
    for (int i=R; i>= 1; i--) {
    B = Integer. rotateRight (B - S[2* i+1], A)^A;
    A = Integer. rotate Right (A -S [2* i], B) ^B;
    }
```

TOPIC NAME : IT-21031 DAY :
TIME : DATE : / /

Sample output :

Encrypted : 7f93d8c2 1423ba29

Decrypted : 12345678 9abcdef0

Explanation :

- The input plaintext is : 0x12345678
  0x9abcdef0
- The encrypted ciphertext looks random
- After decryption we get back the exact original plaintext