Shetu Saha
IT-21009

Assignment-6

Name of Assignment: Modes of Operation and RCB-Block Diagram and Java Implementation and output.

Modes of operations:

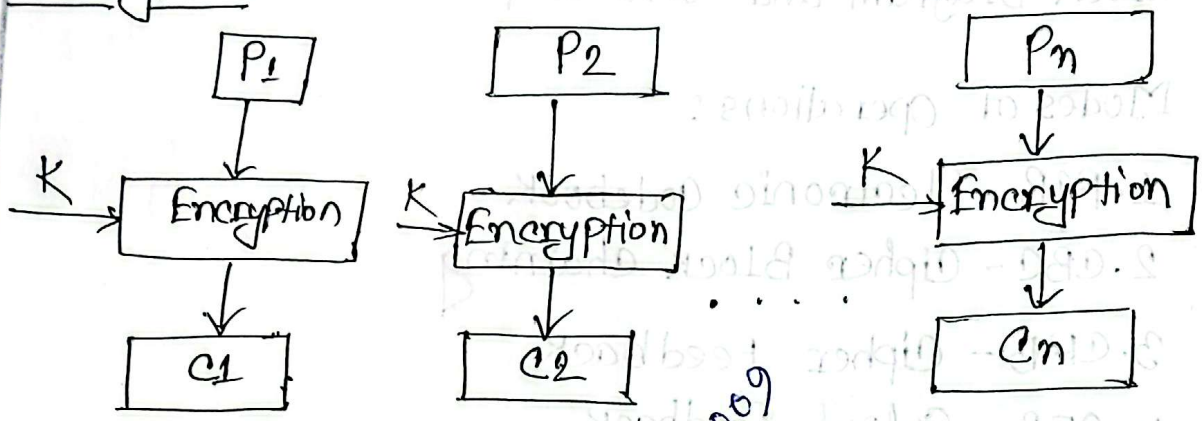
1. ECB - Electronic Codebook
2. CBC - Cipher Block Chaining
3. CFB - Cipher Feedback
4. OFB - Output feedback
5. CTR - Counter

Description:

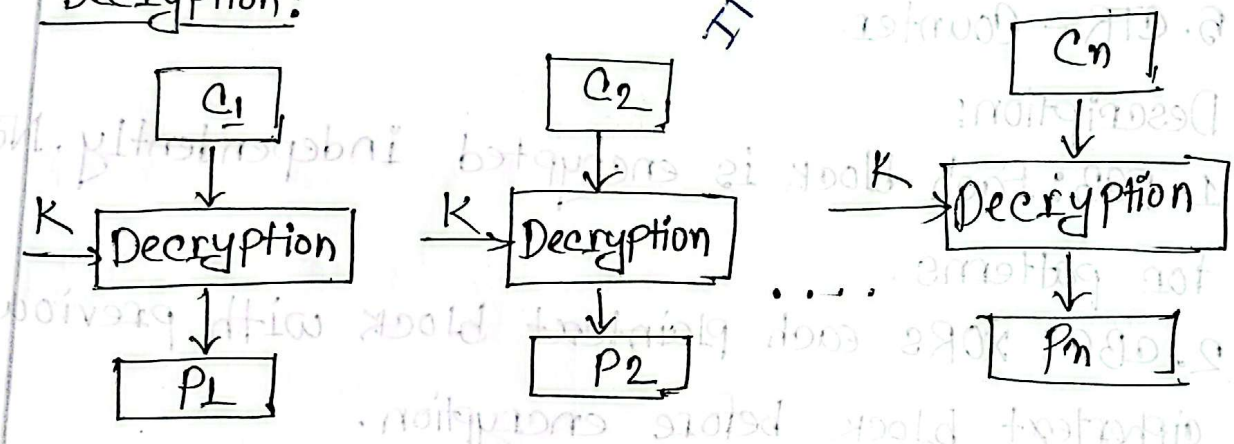
1. ECB: Each block is encrypted independently. Not secure for patterns.
2. CBC: XORs each plaintext block with previous ciphertext block before encryption.
3. CFB: Converts block cipher into a self-synchronizing Stream cipher.
4. OFB: Turns block cipher into a synchronous Stream cipher.
5. CTR: Uses a counter that gets encryted and XORed with plaintext - fast and parallelizable.

The procedure of ECB is illustrated below:

Encryption:

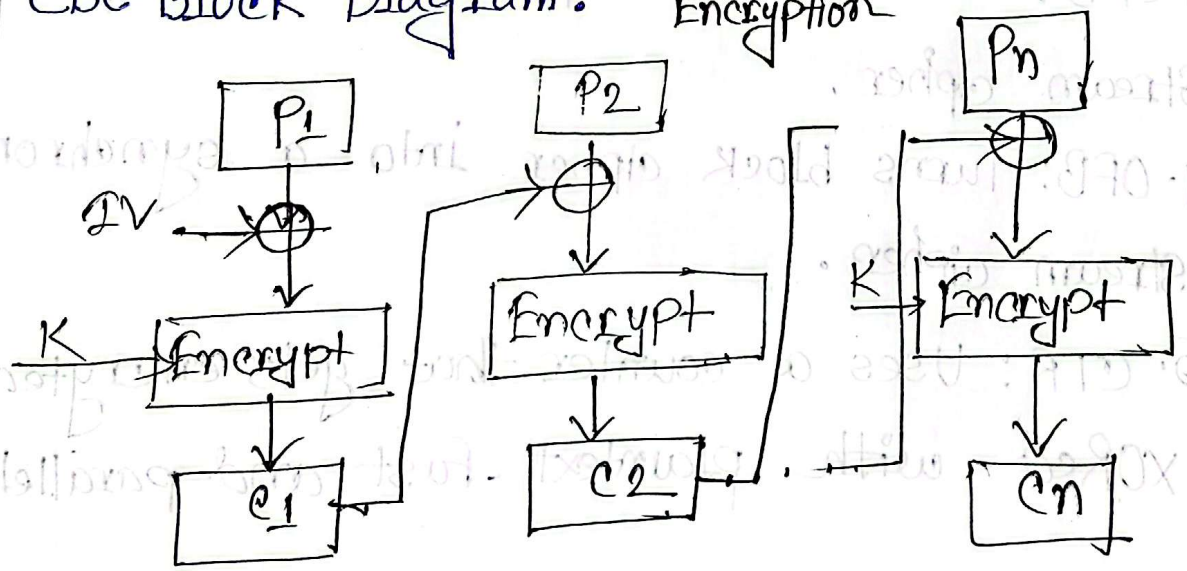


Decryption:

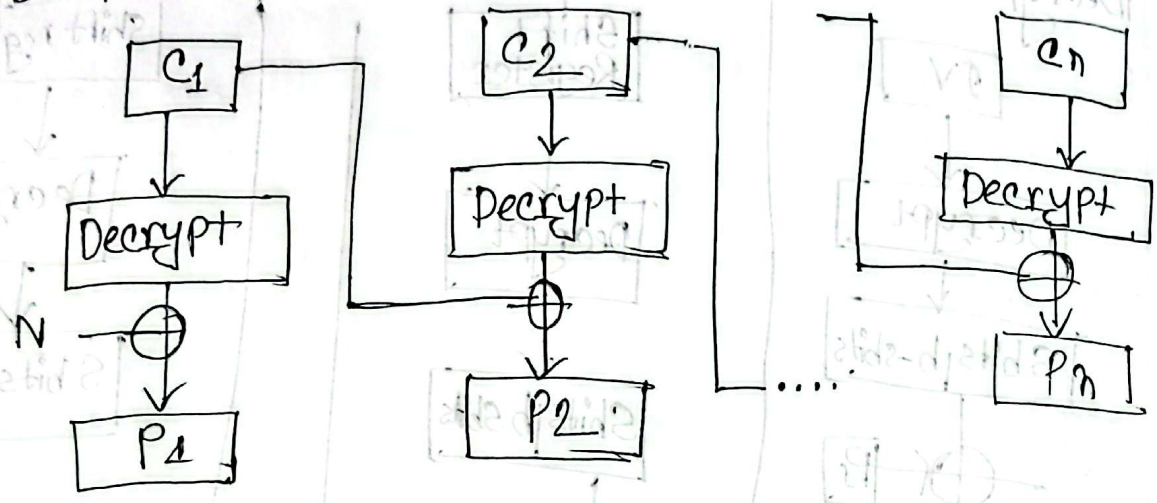


IT-21009

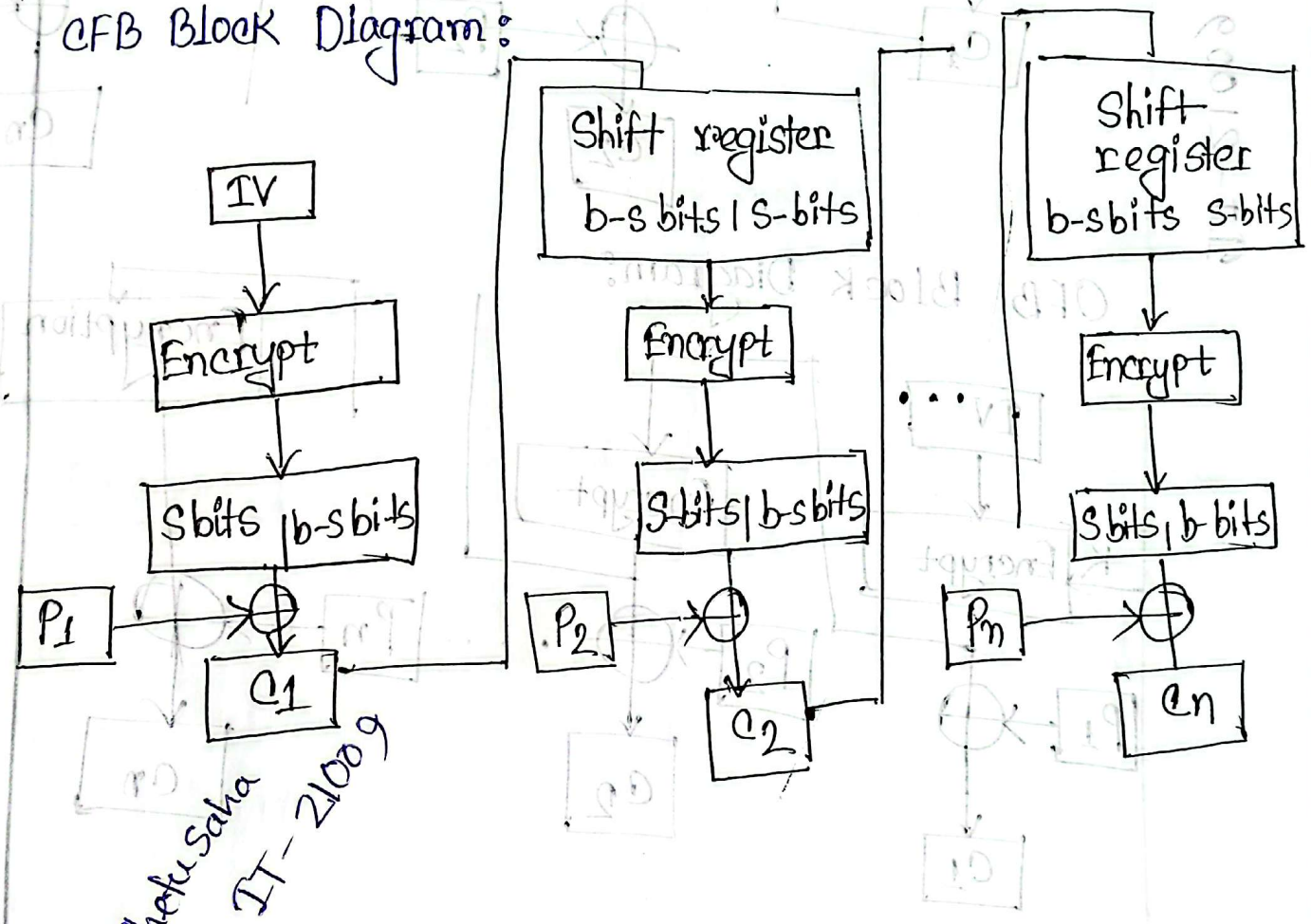
CBC Block Diagram:



Decription :

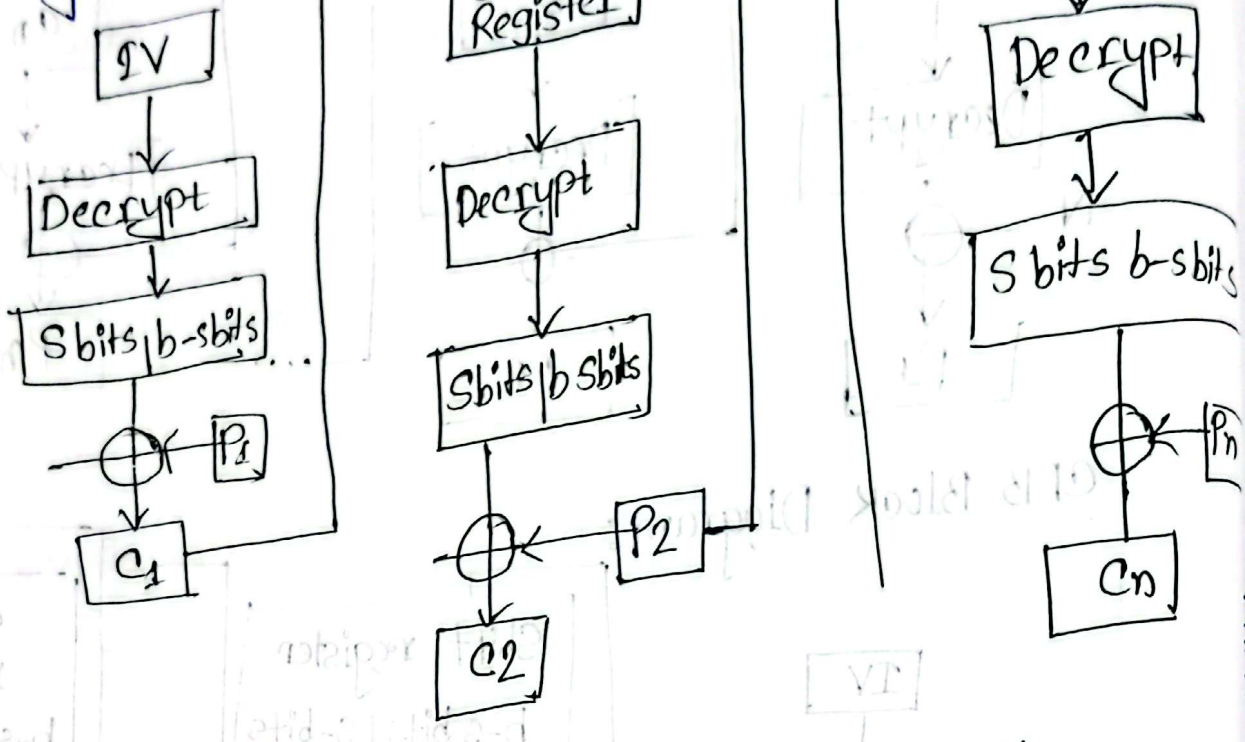


CFB Block Diagram :



Shakusaha
IT-21009

Decryption:



OFB Block Diagram:

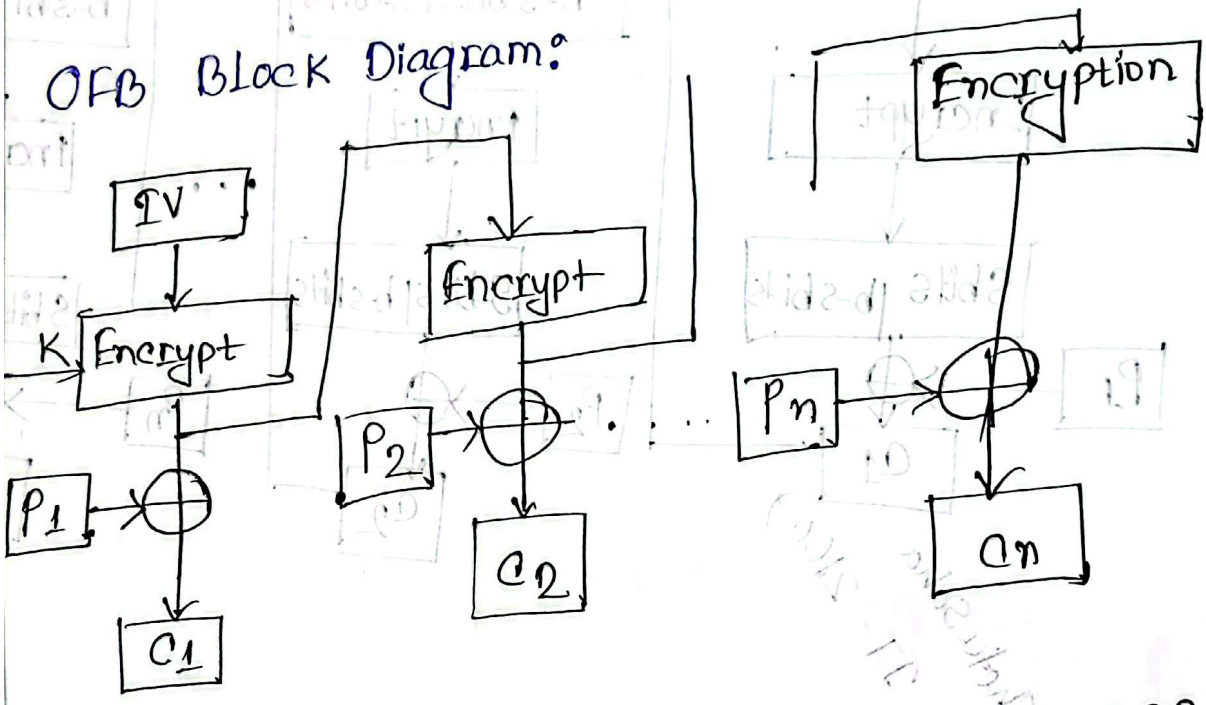
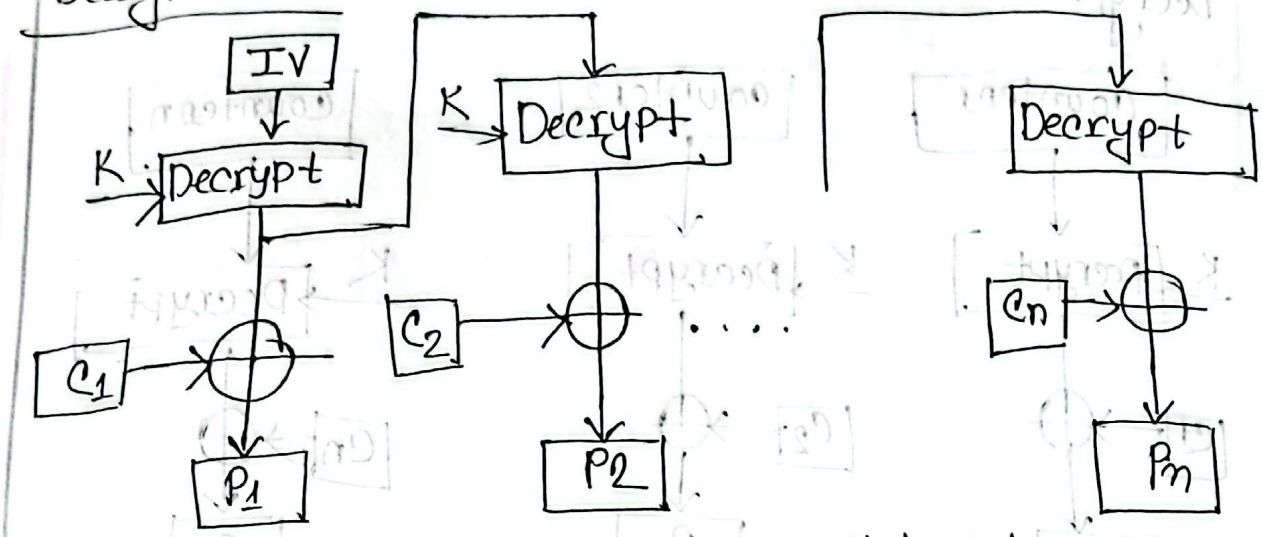


Fig: Block diagram of OFB Encryption

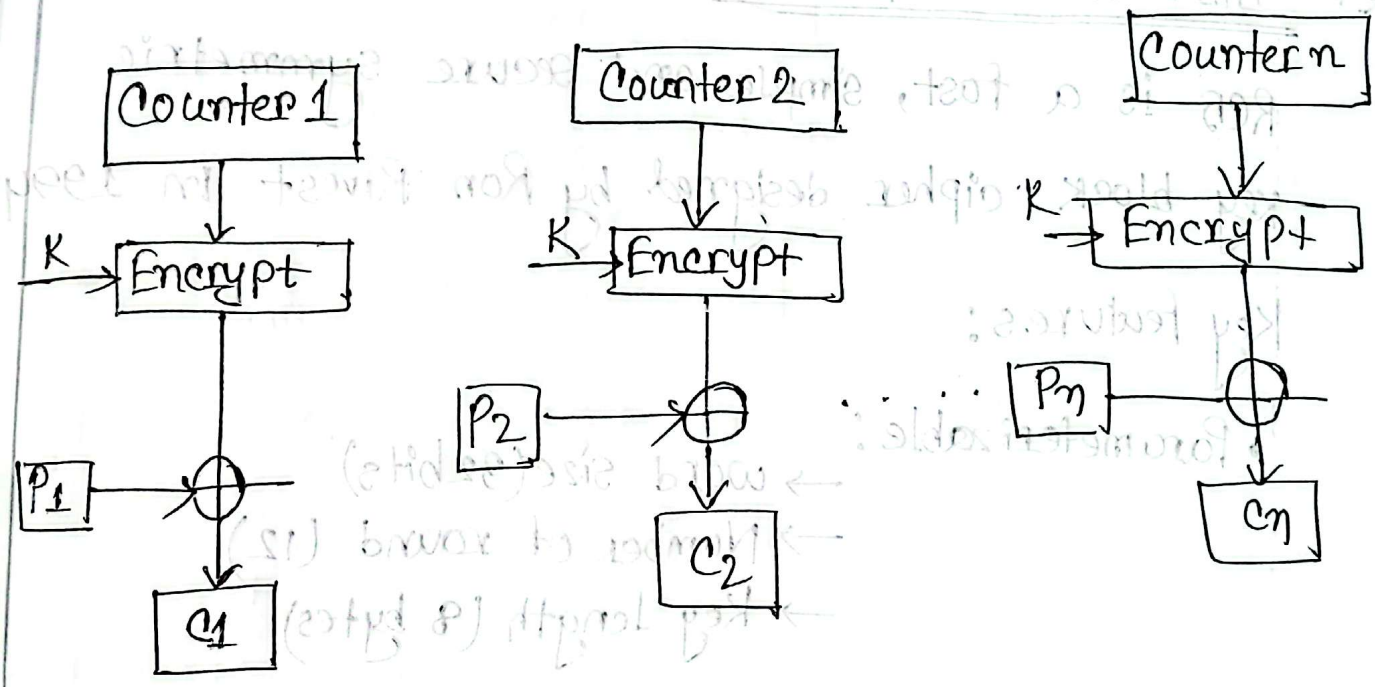
Decryption:



Shetu Saha
II-21009

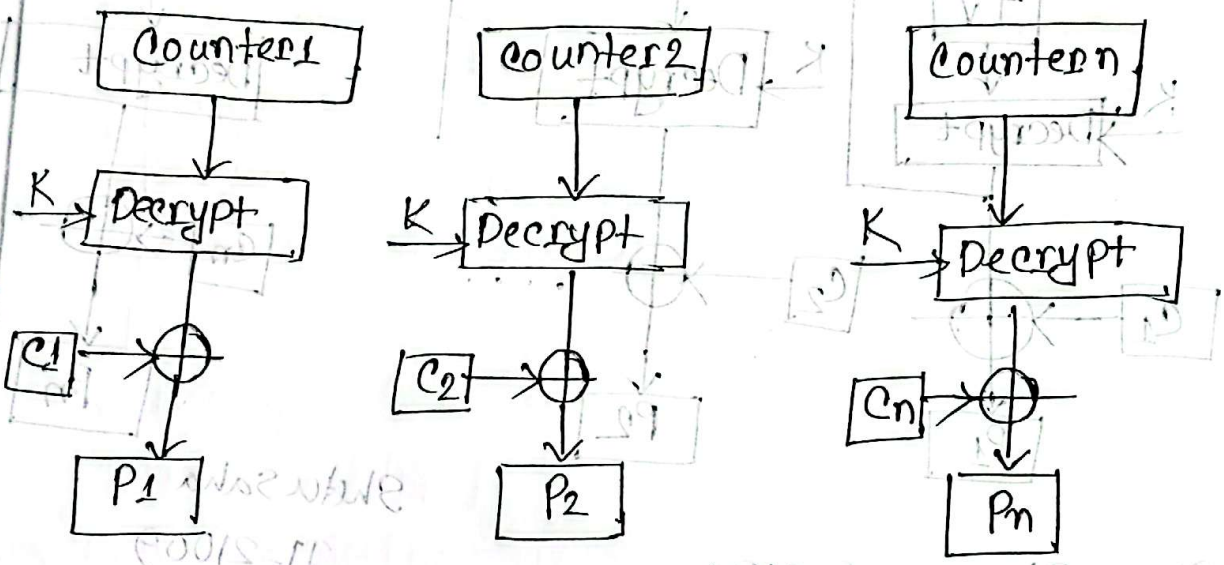
Block Diagram of CTR:

Encryption



IT-21009

Decryption:



Introduction of RC6:

RC6 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, rotate
  - → Modular addition

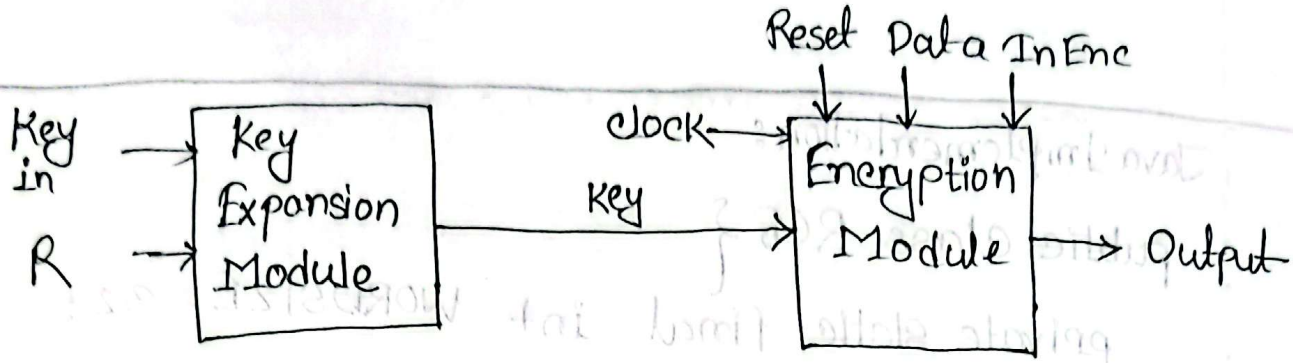
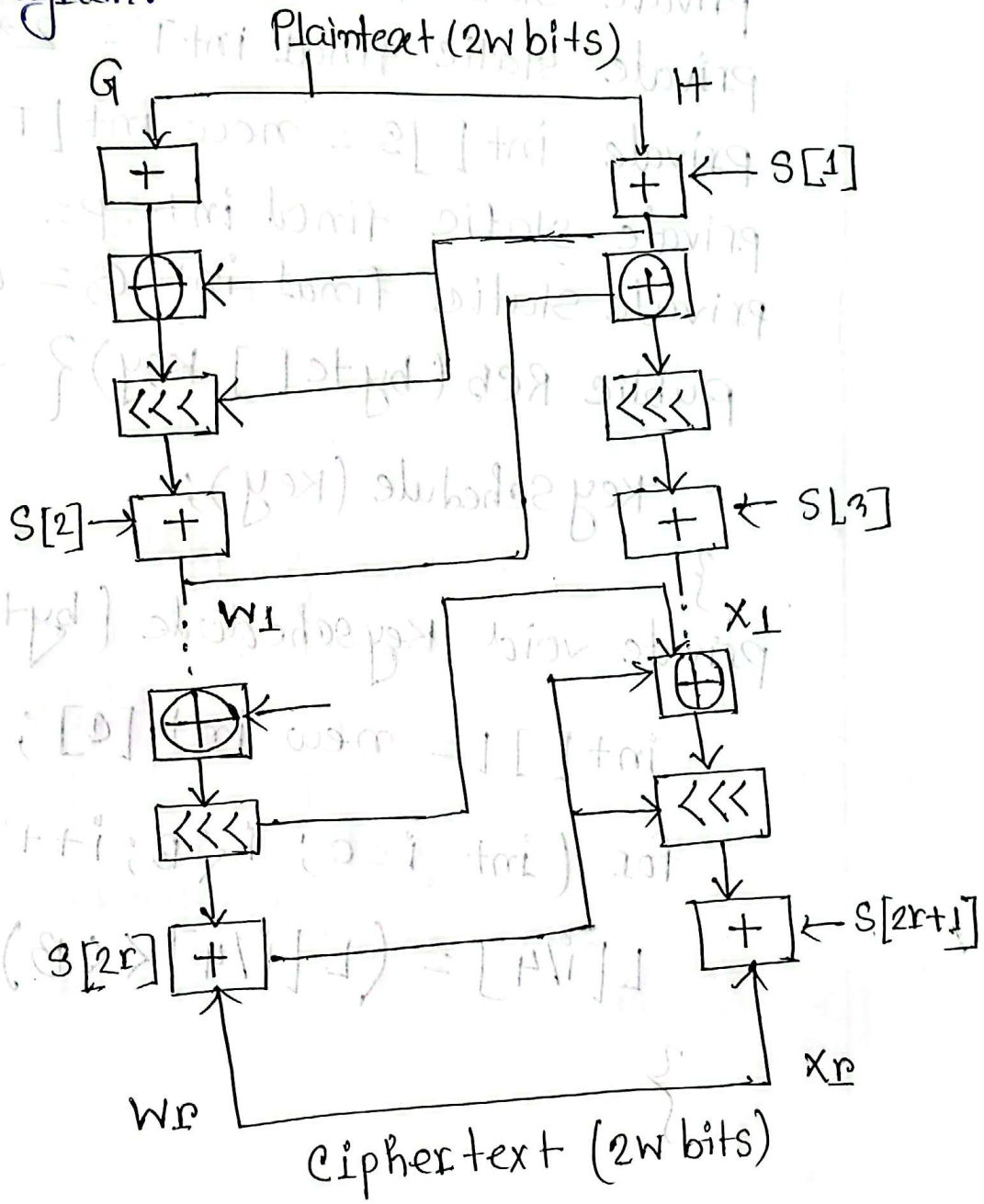


Fig: Block Diagram of RC5 Encryption

RC5 Block Diagram:



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= new int [c];
   for (int i=0; i<B; i++){
     L[i/4] = (L[i/4]<<B)+ (key[i] 20xA
   }
}
```

```
S[0] = P;
for (int i=1; i<T ; i++) {
  S[i] = S[i-1] + Q;
 }
 int A=0, B=0, i=0, j=0;
for (int K=0,K<3* T; K++) {
 A = S[i] = Integer. rotateleft((S[i] + A+B), 3);
 B = L[j] = Integer. rotateleft((L(j)+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];
    }
```

```
    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.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 = "password".getBytes();
RC5.rc5 = new RC5(Key);
int [ ] pt = {0x12345978, 0x9abcdef0};
int [ ] ct = rc5.encrypt(Pt);
System.out.printf("Encrypted : %08X %08X\n",
    ct[0], ct[1]);
int [ ] dt = rc5.decrypt(ct);
```

IT-2109

```
System.out.print("Decrypted : %08x %08x\n", dt[0]
dt[1);
}
}
```

IT-21069

Sample Output :

Encrypted: 7f93d8c2 1423ba29

Decrypted: 12345678 9abedef0