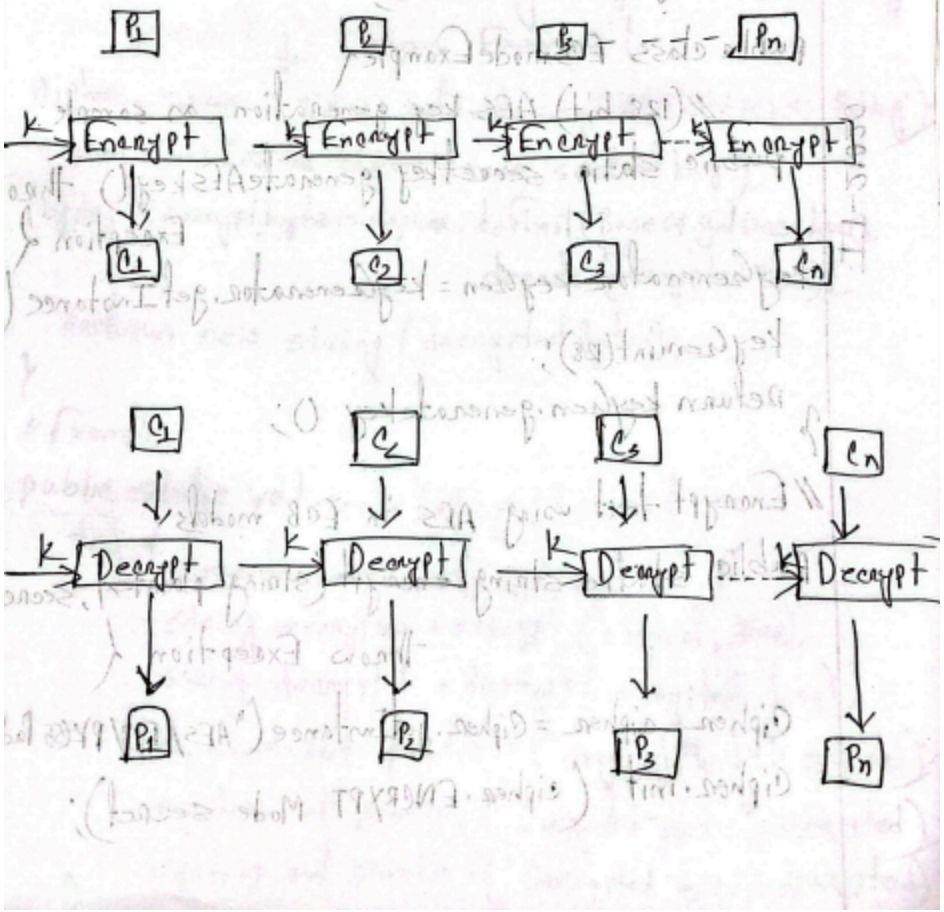
ECB (Electric Codebook): In an, electric codebook each block of bits of plain text is encoded independly with the same key.

Block Diagram:



Java Implementation

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class ECBmodeExample{
    //(128 bit) AES key generation as sample
  public static secretkey generateAESkey() throw
                                        Exception {
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
  keyGen.init(128);
  Return keyGen.generatekey();
}
// Encrypt text using AES in ECB models
public static string encrypt(string plaintext, secretkey)
                        Throw Exception {
  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
  Cipher.init(cipher.ENCRYPT Mode secret);
```

```
byte[] encryptBytes = cipher.doFinal(plainText.getBytes());
  return Base64.getEncoder().encodeToString(encryptBytes);
}
// Decrypt cipher text using AES in ECB ~~using~~ mode
public static String decrypt(String encryptedText, SecretKey
        secretKey) Throws Exception {
 Cipher cipher = Cipher.getInstance("AES/ECB/PCKS5 Padding");
 cipher.init(cipher.DECRYPT_mode, secretKey);
 byte[] decryptbytes = cipher.doFinal(Base64.getDecoder()
                                  decode(encryptedText));
  return new String(decryptedBytes);
}
// Example
public static void main(String[] args) {
   try { secretkey = generateAESkey();
         String encrypted = encrypt(original, key);
         String decrypted = decrypt(encrypted, key);
         System.out.println = ("original Text:" + original);
         System.out.println = ("Encrypted Text:" + encrypted);
         System.out.println = ("Decrypted Text:" + decrypted);
  }
```

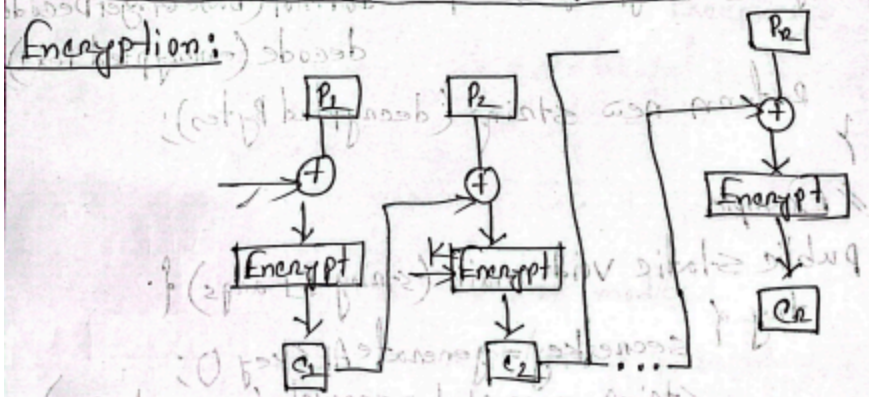
```
catch (Exception e){
    e.printStackTrace();
}
}
```

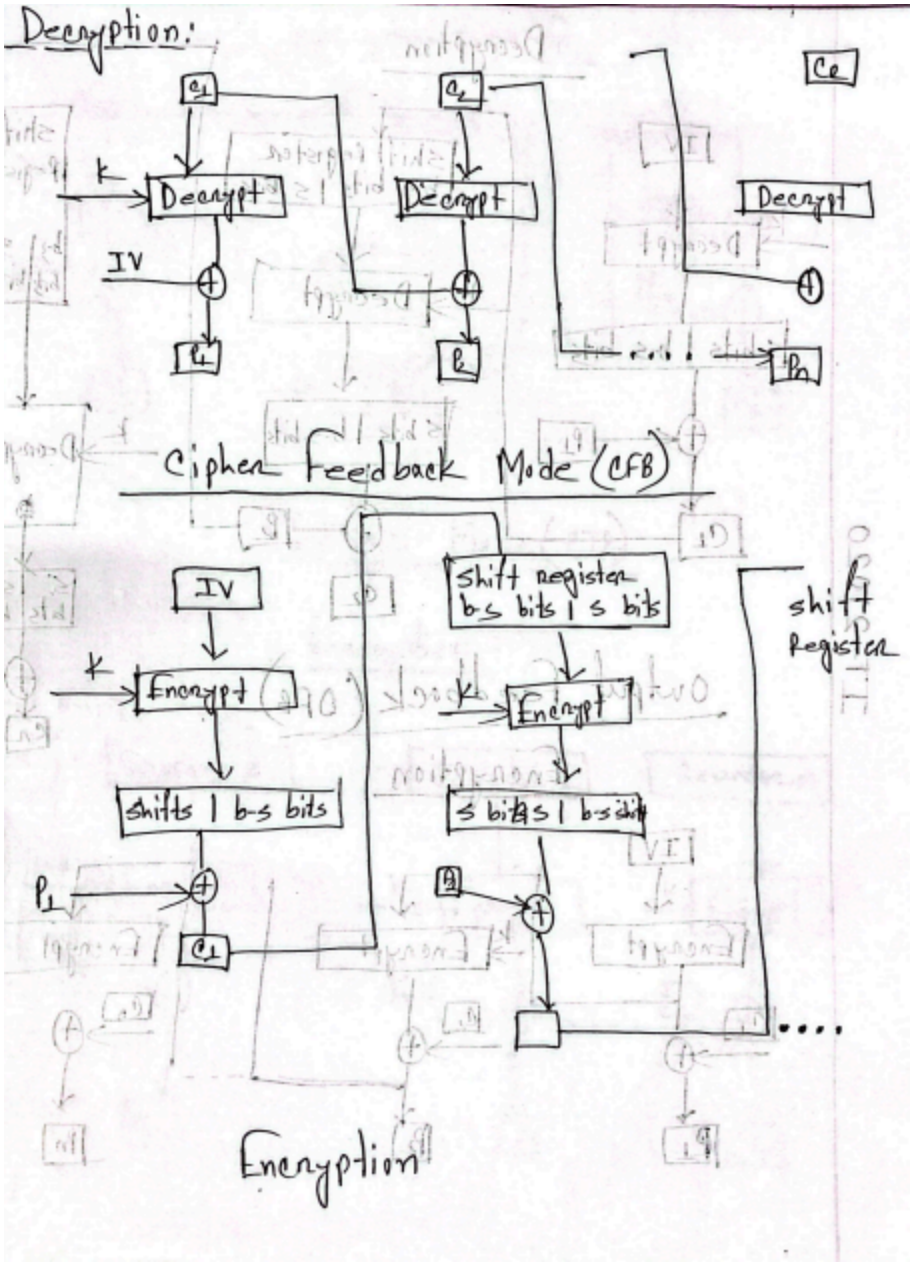
Output

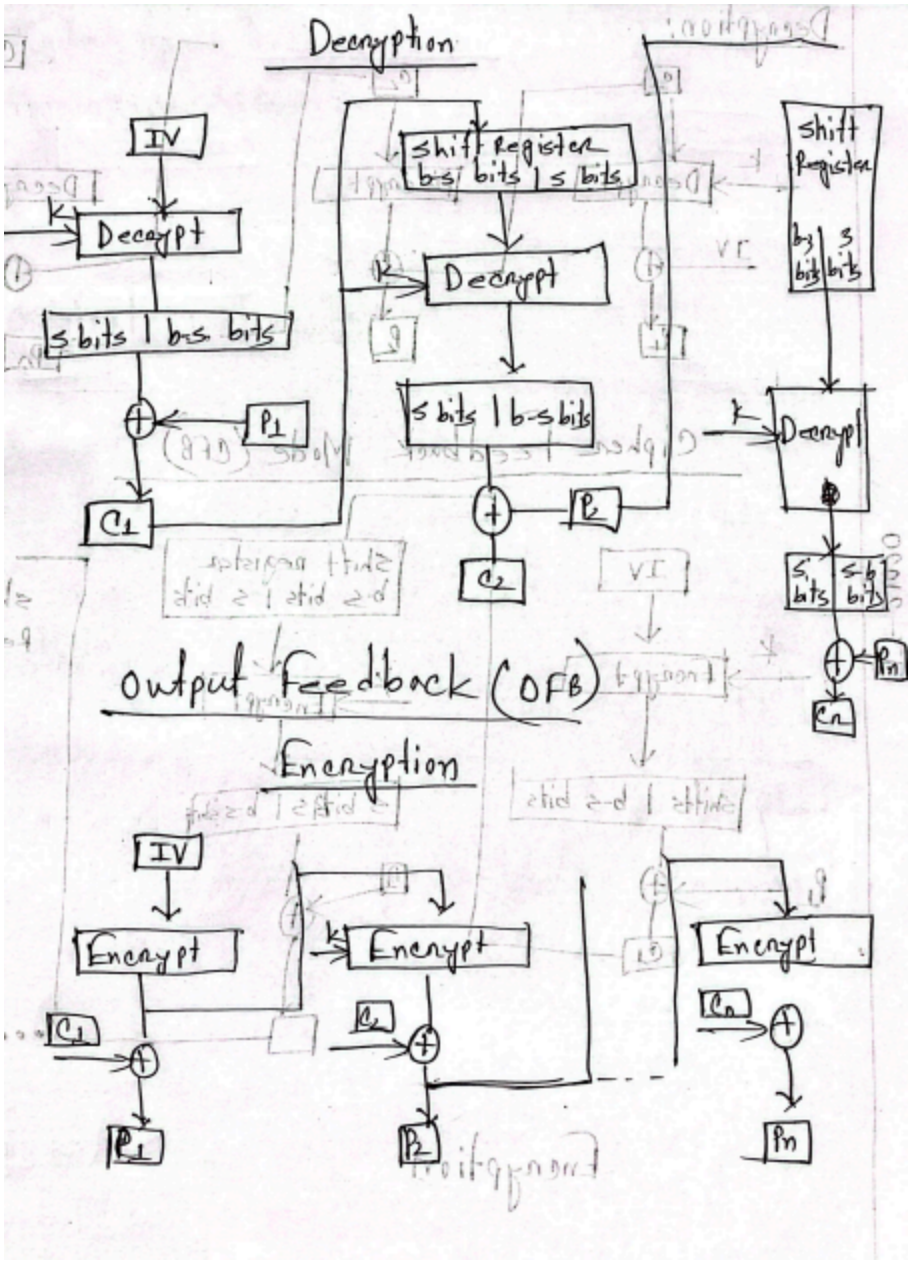
Original Text : Hello cyber world!
Encrypted Text : fRQtHykBP3AscX1vQ1QwP1qnwPgH
Decrypted Text : Hello cyber world!

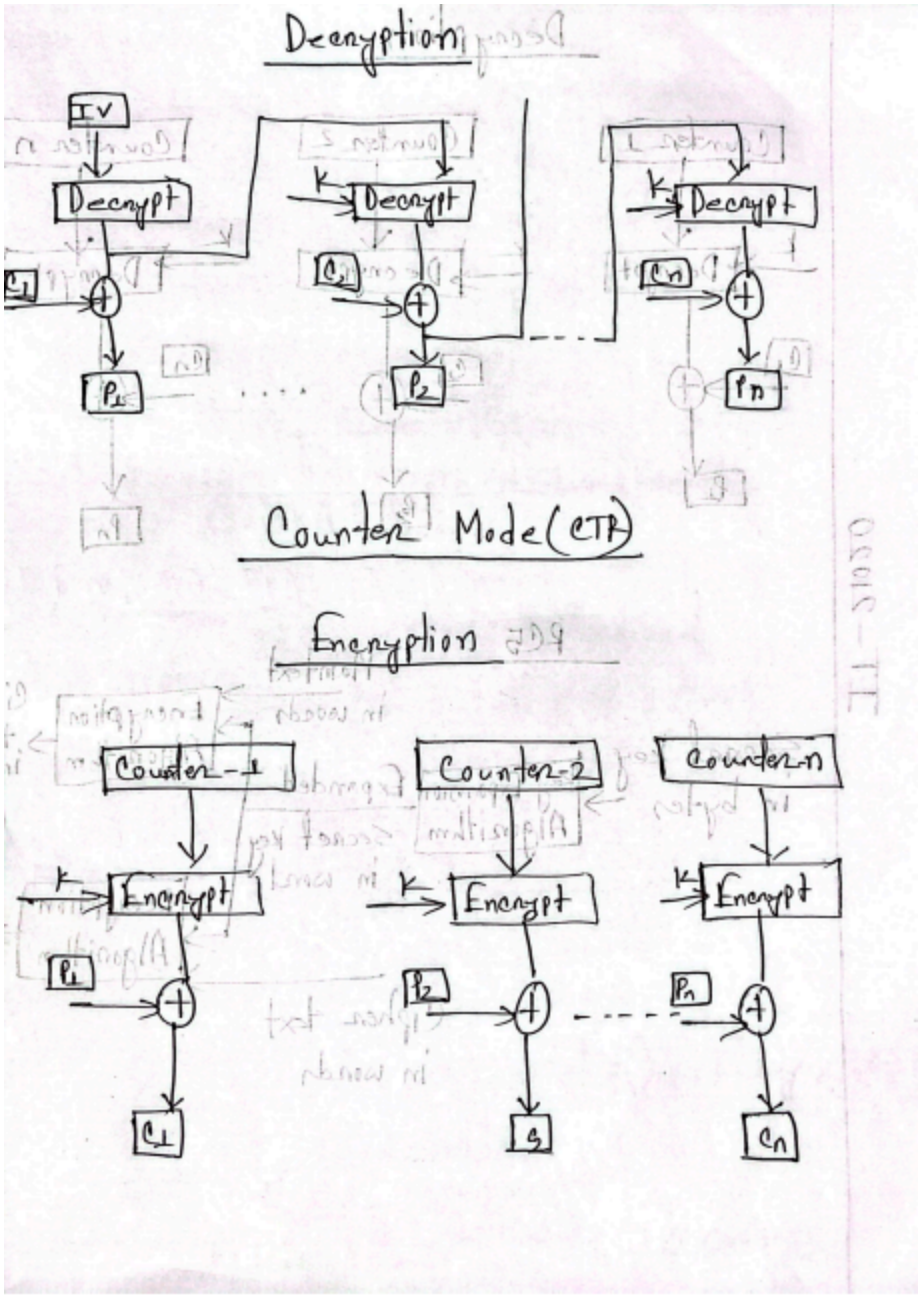
CBC (Cipher Block Chaining)

Encryption:

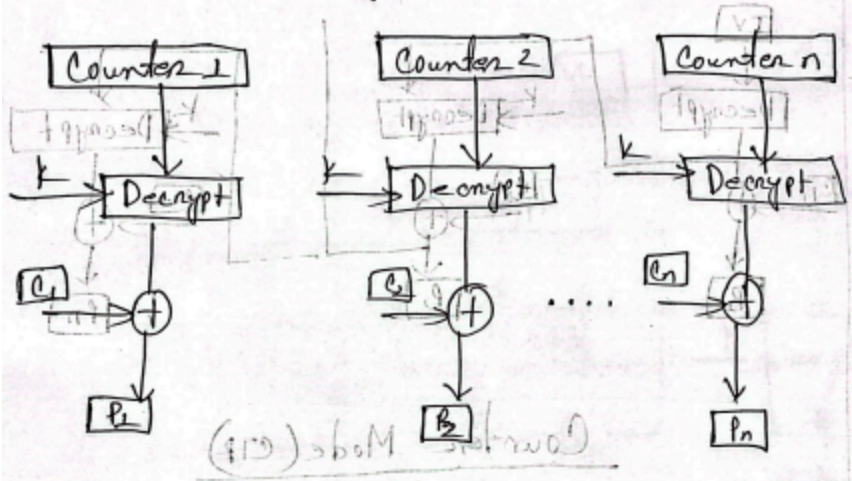



Decryption:
Ciphen Feedback Mode (CFB)
Decrypt
IV
Encrypt
shift register
b-s bits | s bits
shift
Register
Encryption

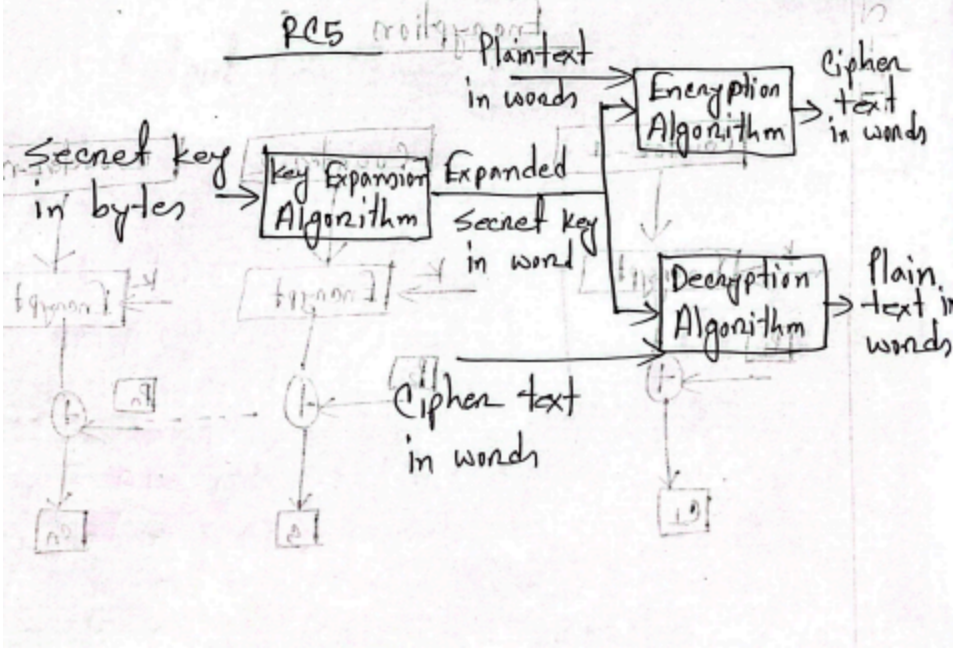

Decryption
IV
Decrypt
shift register
b-s bits | s bits
Decrypt
shift register
s bits | b-s bits
P1
s bits | b-s bits
Decrypt
C1
P2
C2
Output Feedback (OFB)
Encryption
IV
Encrypt
Encrypt
Encrypt
C1
C2
Cn
P1
P2
Pn


Decryption
IV
Decrypt
K
Decrypt
K
Decrypt
C1
C2
Cn
P1
P2
Pn
Counter Mode (CTR)
Encryption
Counter-1
Counter-2
Counter-n
K
Encrypt
K
Encrypt
K
Encrypt
P1
P2
Pn
C1
C2
Cn

Decryption



RC5



Code:

```
Package org.example.rc5fx
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;

Public class RC5x1 extends Application {
    Private static final int P = 0XB7E15163;
    Private static final int Q = 0X9E3779B9;

Private static int rotl(int x, int y) {
    Return (x<<y) | (x>>>(32-y));
}

Private static void rc5keysetup(byte[] key, int[])
{ int[] L = new int[c];
    for(int i= B-1; i>=0; i--) {
        L(i/4) = (L[i/4]<<8) + key[i] & 0XFF);
}
```

```
S[0] = P;
for(int i=1; i<2*(R+1); ++i){
    S[i]=S[i-1]+Q;
}
int A=0, B=0;
int i=0; J=0;
for(int k=0; k<3*MathMax(2*(R+1),C); ++k)
    A=S[i]=Rot1((S[i]+A+B),3);
    B=L[j]=Rot1(L[j]+A+B,(A+B)%W); i=(i+1)%(2*(R+1));
    J=(J+1)%C;
}
Private static void RC5Encrypt(int[]S, int[]data){
    int A=data[0];
    int B =data[1];
    A=A+S[0];
    B=B+S[1];
    for(int i=1; i<=R; ++i){
        A=Rot1((A^B),B)+S[2*i];
        B=Rot1((B^A),A)+S[2*i+1];
    }
```

```
RC5 keysetup(key s);
byte[] plainBytes = PlainText.getBytes(standard charsets UTF8
int paddedLenght = (plainBytes.Lenght+7)/8)*8;
byte[] Padded = Array.Copy of(PlainBytes, Padded length);
String Butter ciphertext=new StringBuilder(
    ByteButter. butter= ByteButter wrap(Padded);
  while (butter.hasRemaining() {
    int A = butter.getInt();
    int B=butter.getInt();
    int[] data = {A,B}
    RC5 Encrypt(s, data);
  ciphertext.append (string.format("%.08x %.08x data[0],
                                   data[1]));
  }
  Return cipher.tostring().trip();
}
```