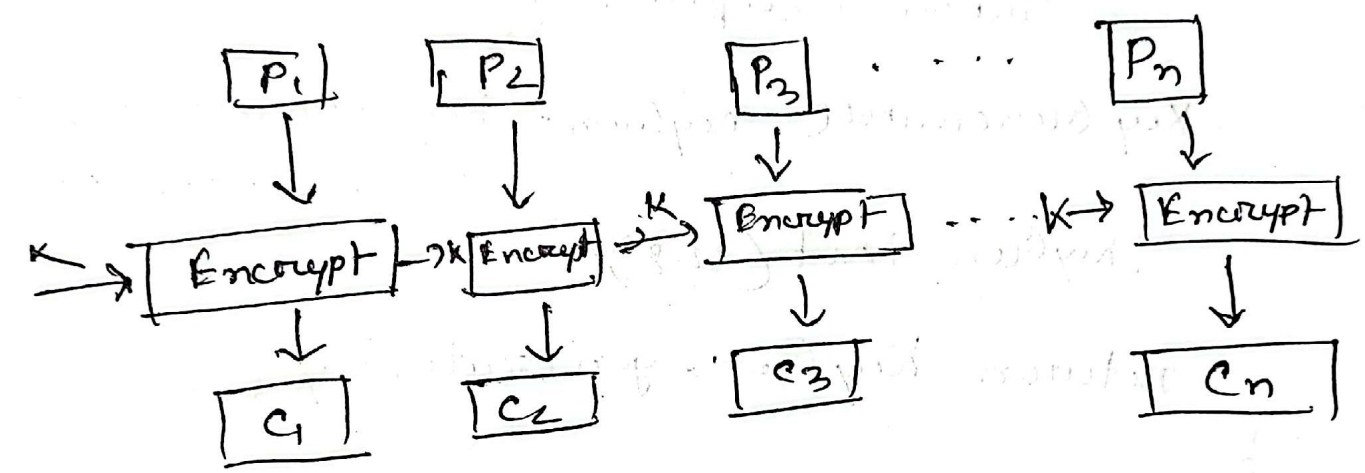
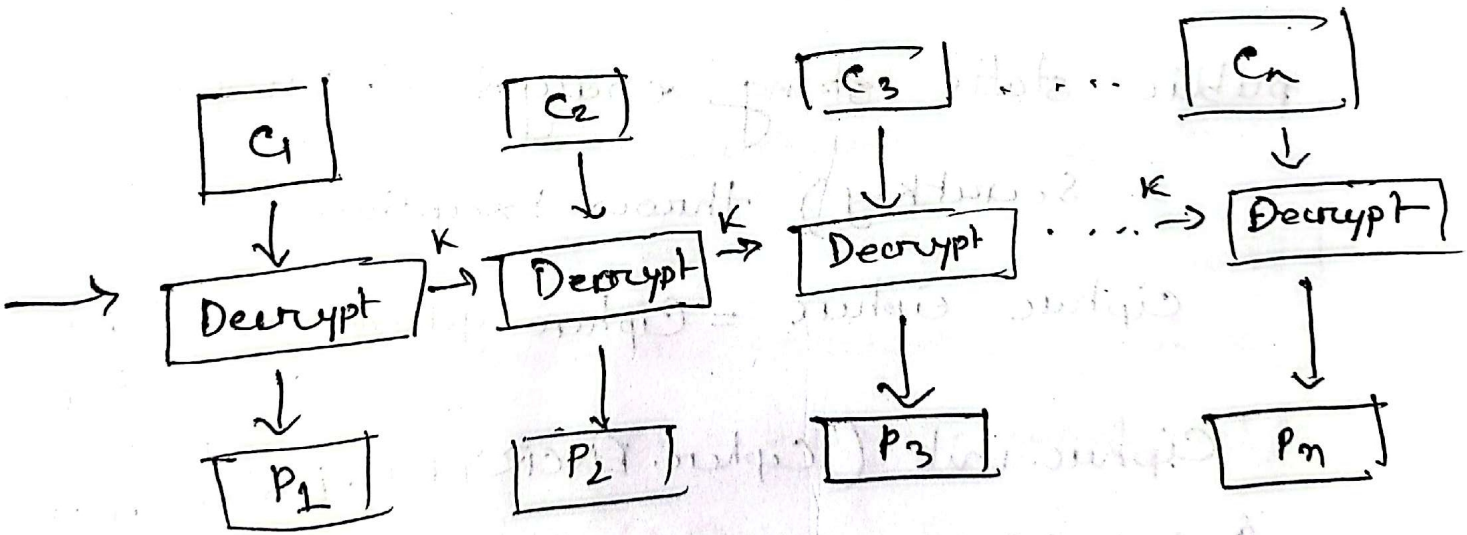
Assignment

IT-21026

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

Block Diagram:





Java Implementation:

```
import javax.crypto.cipher;
import javax.crypto.Keygenerator.
import javax.crypto.*;
```

```
import java.util.Base64;

public class ECBmodeExample {

    public static SecretKey generateAESKey() throws Exception {
        KeyGenerator KeyGen = KeyGenerator.getInstance("AES");
        KeyGen.init(128);
        return KeyGen.generateKey();
    }

    public static String encrypt(String plaintext, SecretKey) throws Exception {
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5padding");
        Cipher.init(Cipher.ENCRYPT_MODE, secret);
        byte[] encryptedBytes = cipher.doFinal(plaintext.getBytes());
        return Base64.getEncoder().encodeToString(encryptedBytes);
    }
```

```
public static string decrypt (string encryptedText,
    SecretKey SecretKey) { throws Exception }
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5
                                   padding");
Cipher.init(Cipher.DECRYPT_Mode, SecretKey);
Byte[] decryptedBytes = cipher.doFinal(Base64.get
                                       Decoder).
                        decode(EncryptedText);
return string (decryptedBytes);
}
public static void main(String [] args) {
    try {
        SecretKey = generateAESKey ();
        String original = "Hello cyber world!";
    String encrypted = encrypt (original key);
    String encrypted = decrypt (encrypted, key);
    System.out.println (" Original Text:" + original);
                println (" Encrypted Text: " + encrypted);
                println (" Decrypted Text :" + decrypted);
    }
catch (Exception) { e.printstackTrace(); }
}
```

Output :

IT-29026

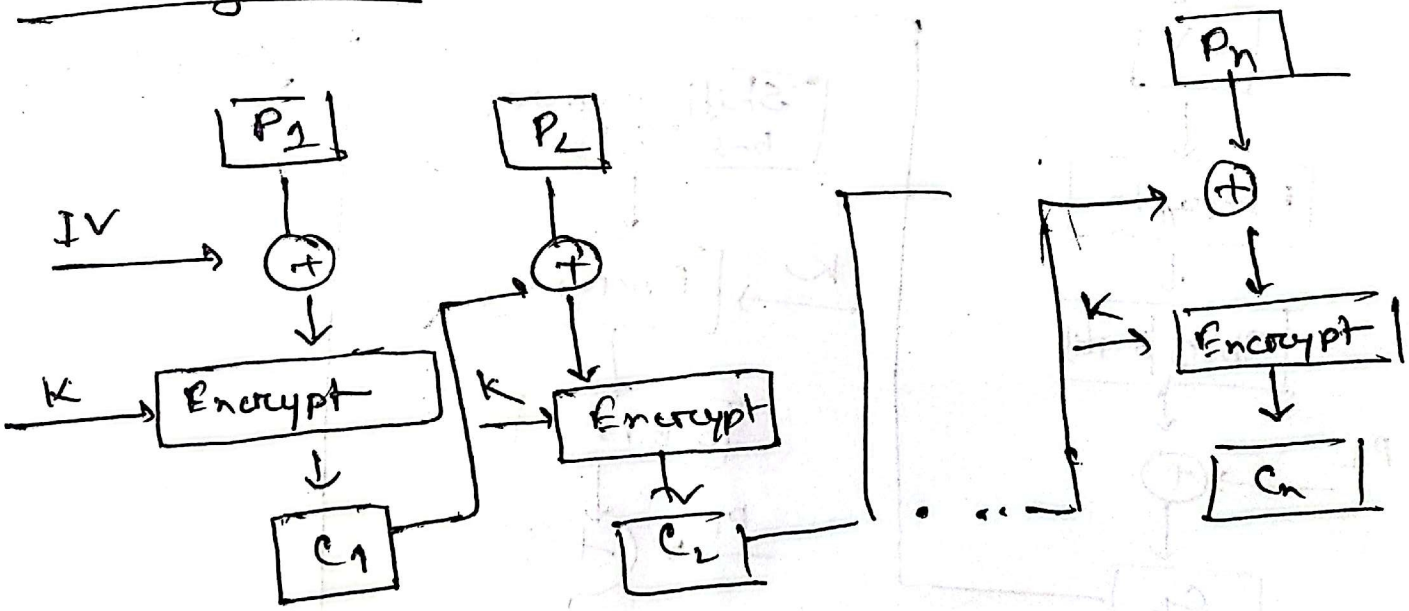
Original Text: Im Nasimul Hasan

Encrypted Text: JEOgB1X2T2a6Q16hvQa+A
h29Mj/YY2810NxWKB2JJgK==

Decrypted Text: Im Nasimul Hasan

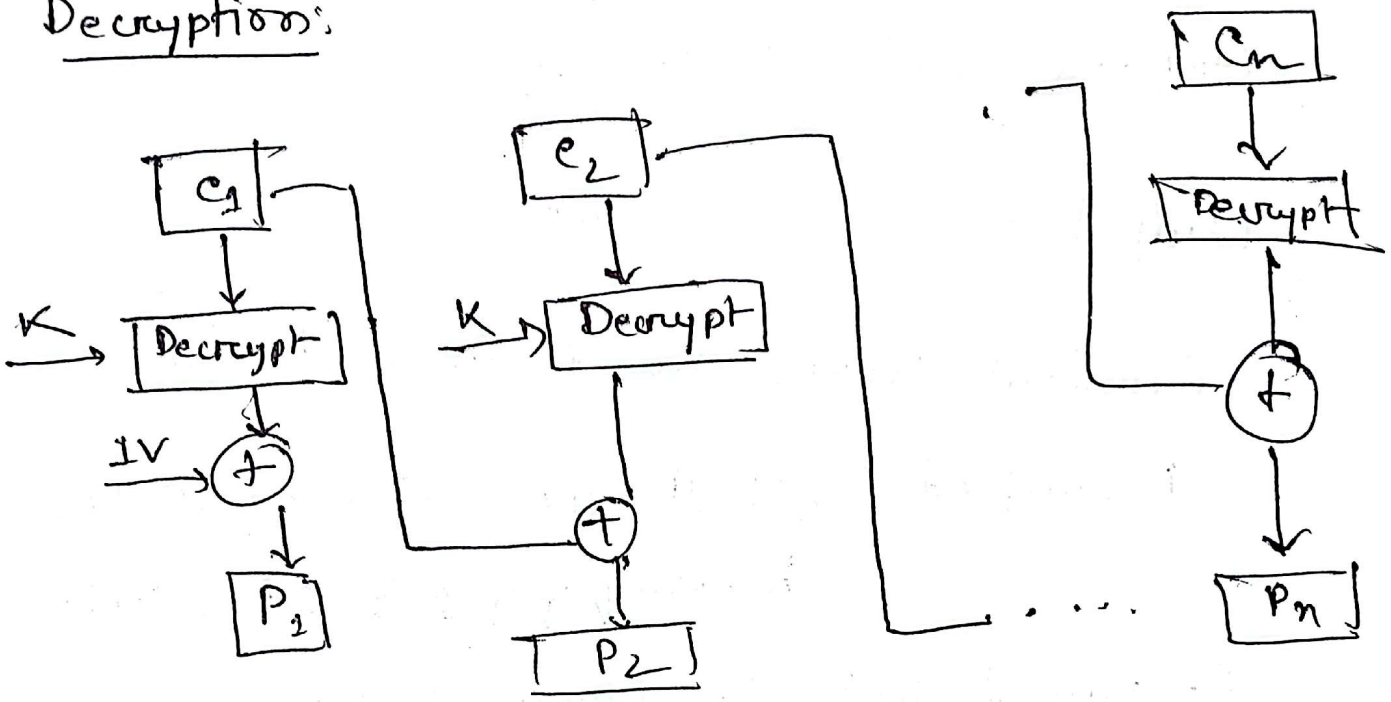
CBC (Cipher Block Chainig)

Encryption



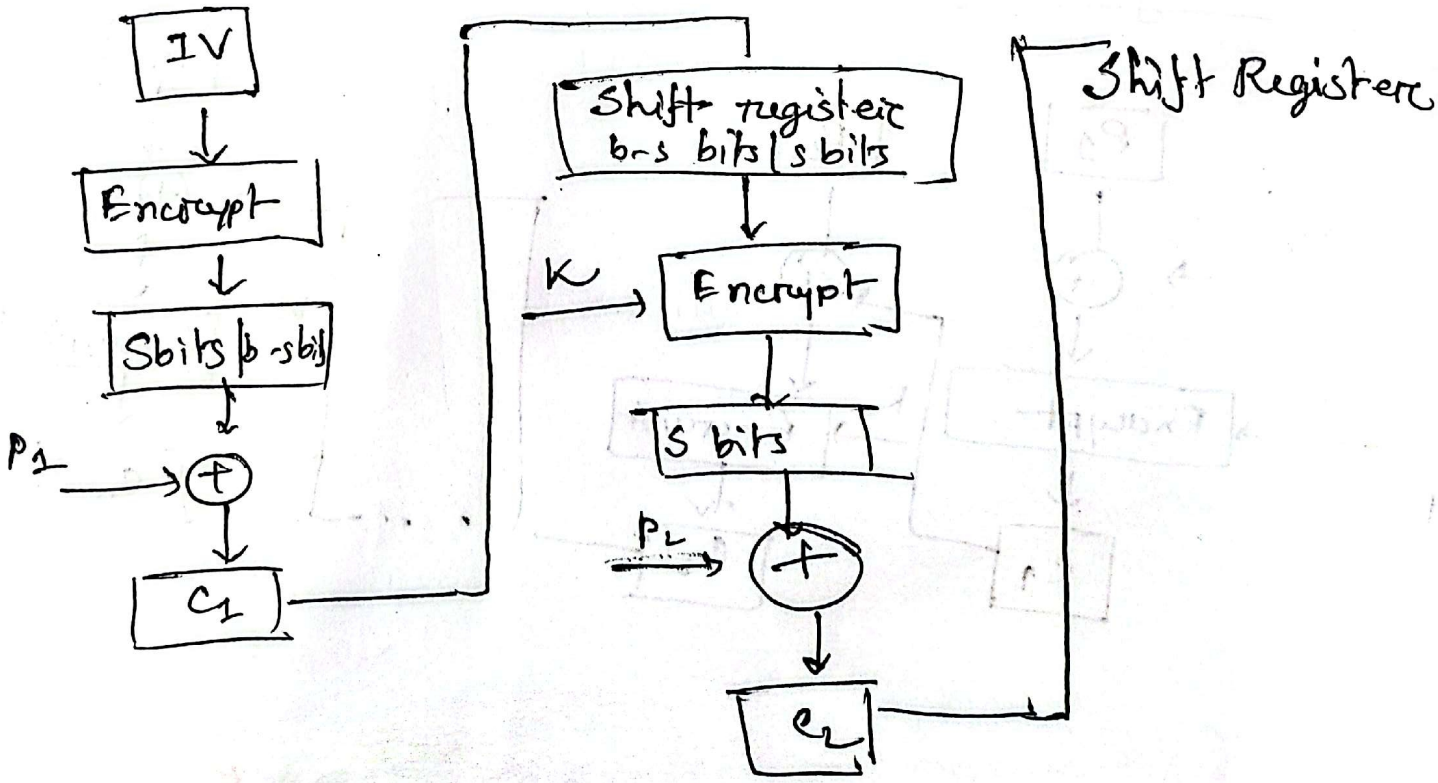
IT-21026

Decryption:

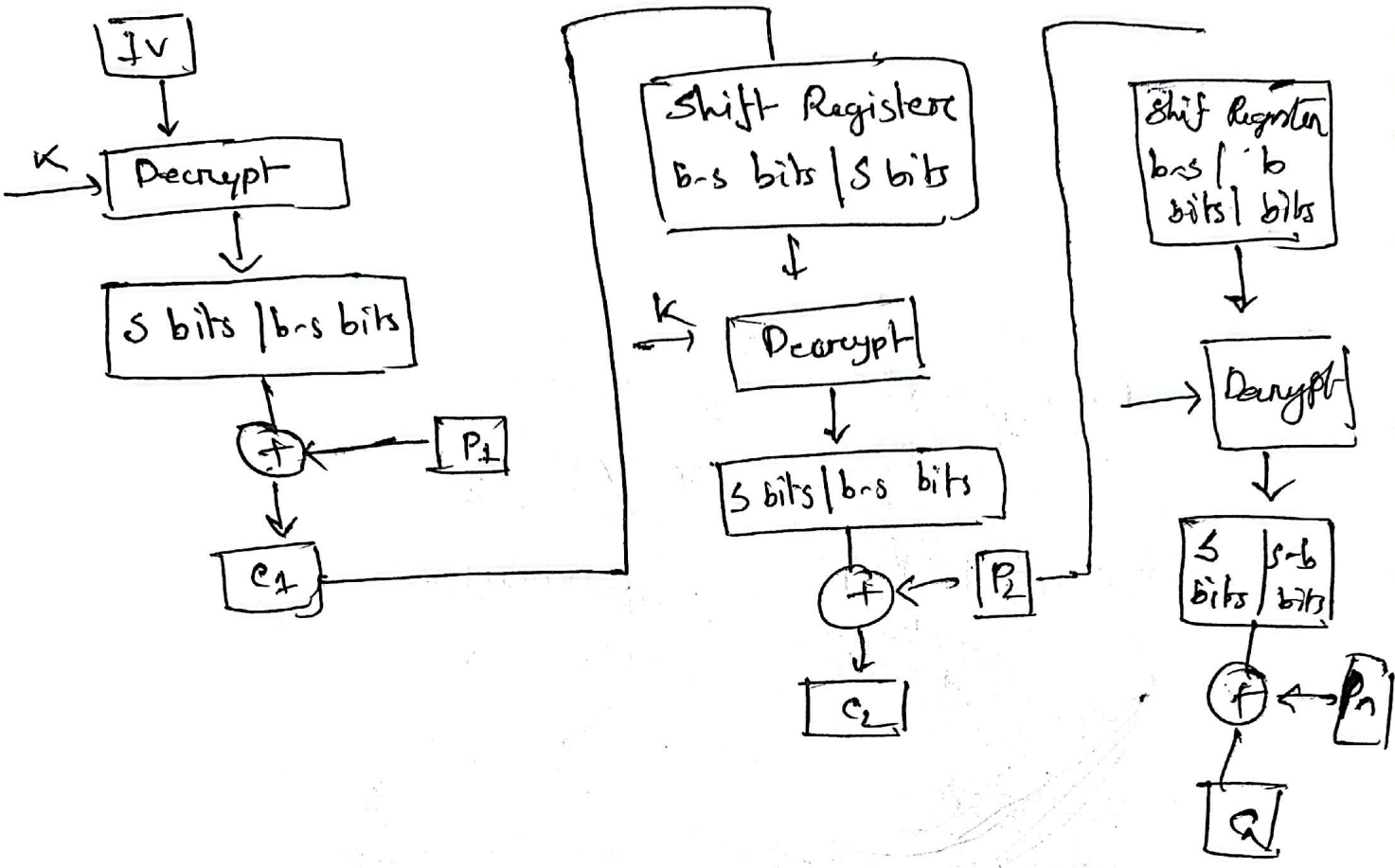


Cipher Feedback Mode (CFB)

Encryption

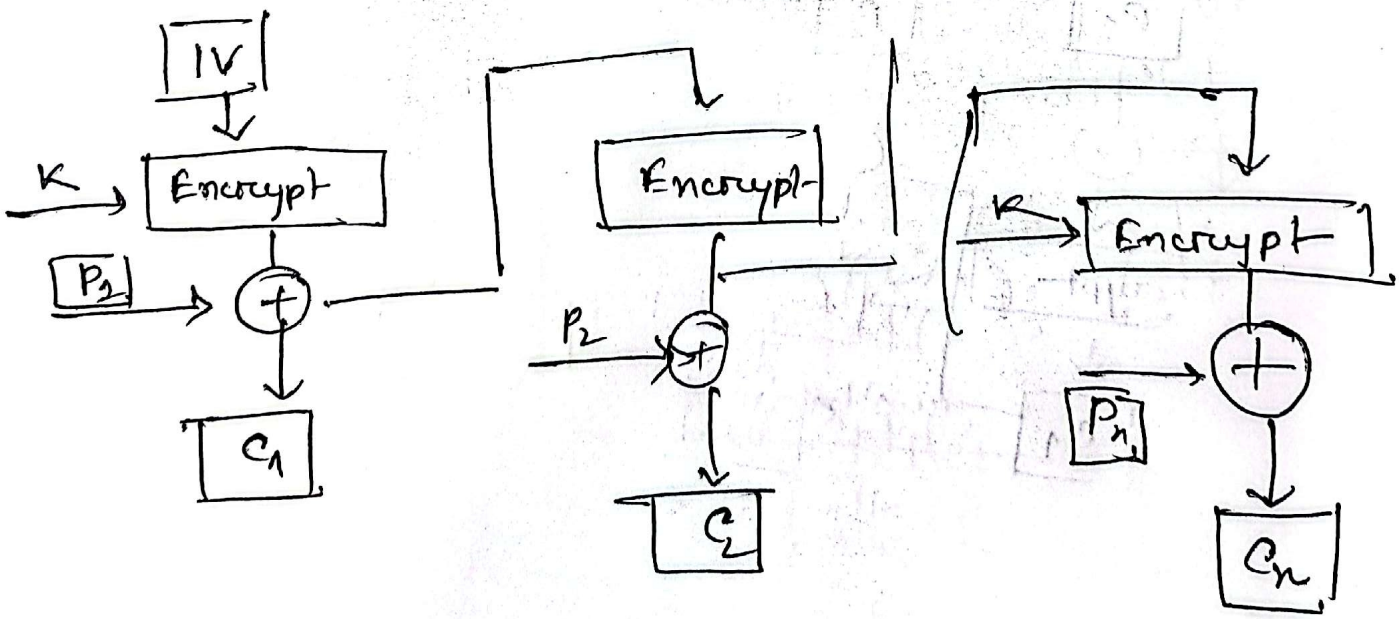


Decryption



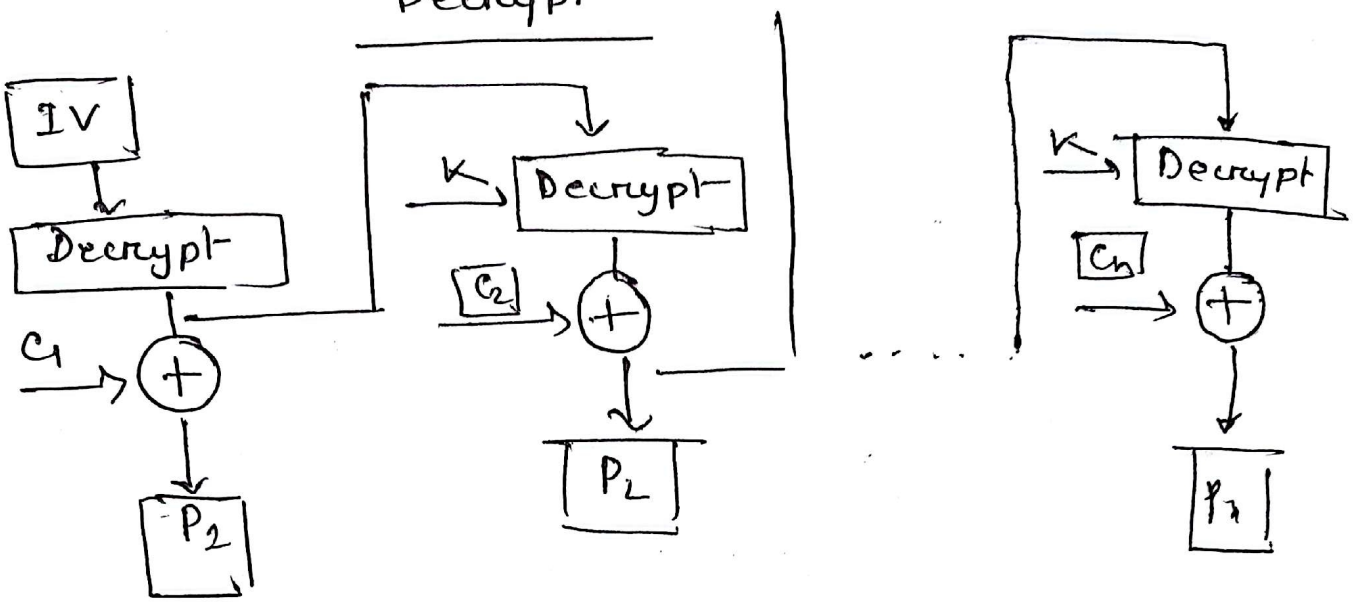
Output Feedback (OFB)

Encryption



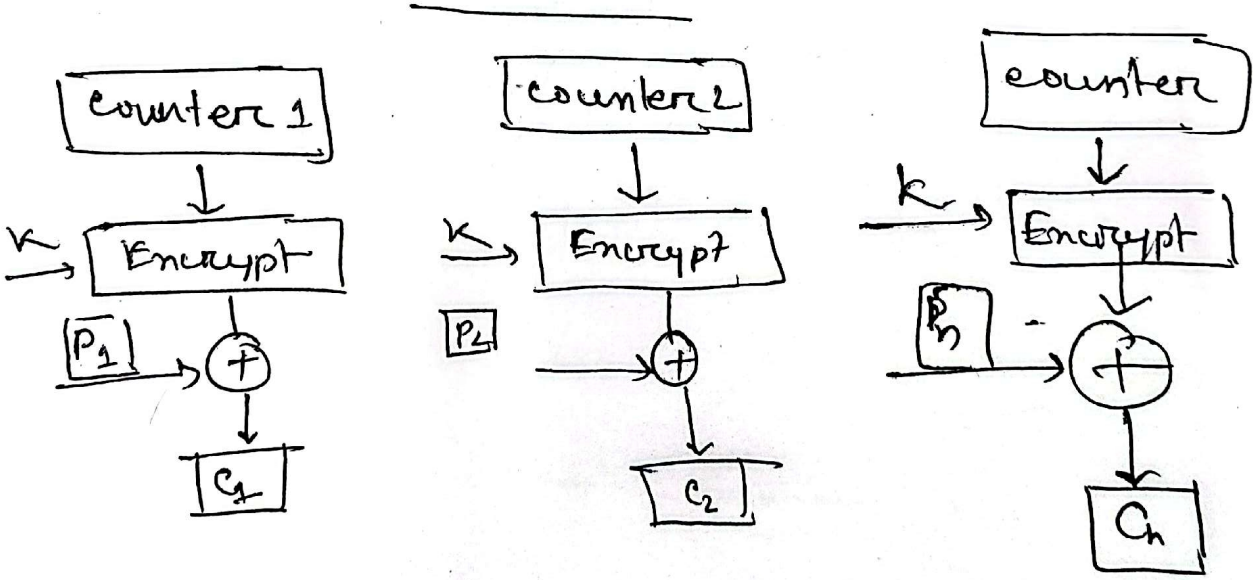
IT-21026

Decrypt



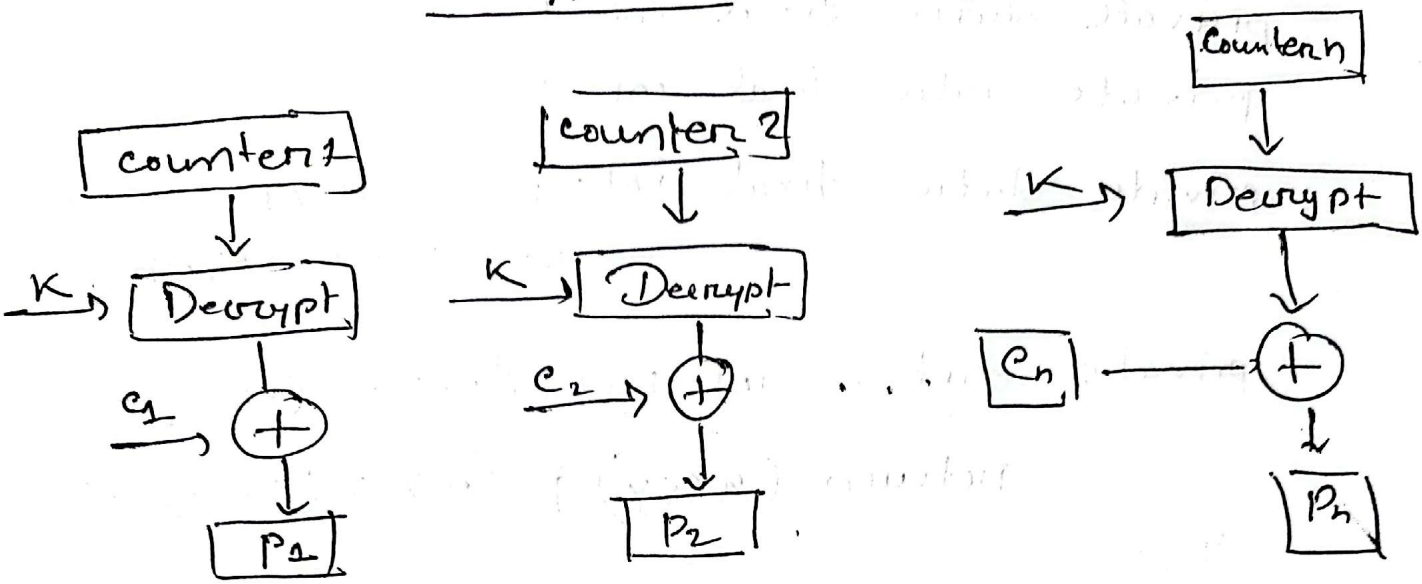
Counter Mode (CTR)

Encryption

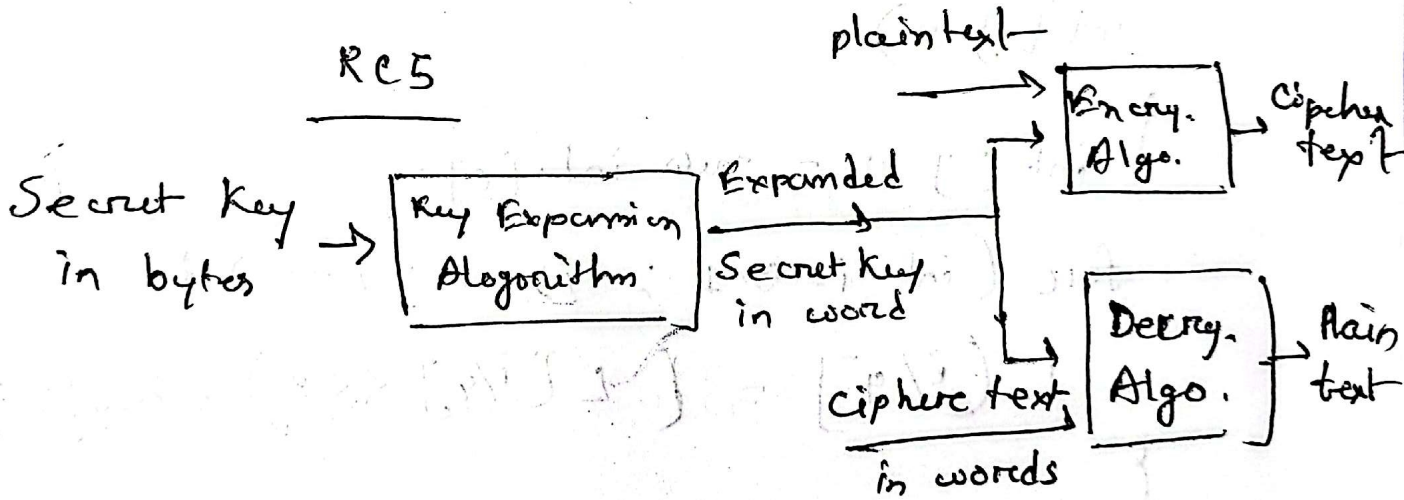


Decryption

IT-24026



RC5



Code:

```
package org.example.rc5fx;
import javafx.*;
import java.*;

public class RC5FX1 extends Application {
    private static final int w = 32;
```

```
private static final int R = 12;
private static final int B = 16;
private static final int P = 0xB7E15163;
                     Q = 0x9E3779B9;

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

private static void rc5KeySetup (byte[] key,
  int[] s)
{
  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 = 0; i<2*(R+1); ++i) {
        s[i] = s[i-1] + B;
}
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] = rotl (S[i]+A+B, 3);
    B= L[j] = rotl (L[j]+A+B, (A+B)%, 4);
     i = (i+1) %. (2*(R+1));
     j = (j+1) %c ;
    }
}
private static void RC5 Encrypt (int[] s, int[] data)
{
    int A = data[0];
    int B = data [1];
    A = A+S[0];
    B = B+S[1];
  for (int i=1 ; K=R; ++i){
        A= rotl (A^B, B) + S[2*i];
        B = rotl(B^A, A) + S[2*i+1];
    }
    data[0] = A;
}   data[1] = B;
```

IT-21026

```
private static string encryptText (String plainText){

    byte[]Key = new byte [8];
    int [] s = new int [2*(R+1)];

    rc5 KeySetup (Key, S);

byte [] plainBytes = plainText.getByte (StandardCharacters.UTF-8);

int paddedLength = (((plainBytes.length + 7)/8) * 8);

    byte [] padded = Array.copyOf (plainBytes, paddedLength);

string Builder ciphertext = new StringBuilder ();
Byte Buffer buffer = ByteBuffer.wrap (padded);
while (buffer.hasRemaining()) {
    int A = buffer.getInt(),
        B = buffer.getInt();
    int [] data = {A, B};
    rc5 Encrypt (s, data);
ciphertext.append (string.format("%.08x %.08x", A, B));
```

```
return ciphertext.toString().trim();
}
@Override
public void start(Stage stage) {
TextField input = new TextField();
input.setPromptText("Enter English word to
                     encrypt");
Button encryptButton = new Button("Encrypt");
    TextArea output = new TextArea();
    output.setEditable(false);
    output.setWrapText(true);
encryptButton.setOnAction(e -> {
    String plainText = input.getText();
if (plainText != null && !plainText.isEmpty()) {
    if plain String ciphertext = encryptText(plainText);
}
else {
    output.setText("Please Enter Some Text");
}; }
```

IP-24026

```
VBox layout = new VBox (10, input, encryptBtn,
                        output);
layout.setStyle("-fx-padding: 20;");

Scene scene = new Scene(layout, 500, 300);
stage.setTitle("RC5 Encryption Tool");
stage.setScene(scene);
stage.show();
}

public static void main(String[] args){
    launch(args);
}
}
```

Output : Input : 5 M Jisan

padding into 64 bit block.

Block 1 : "5M Jisan" (8 bytes)

Block 2 : "nll|0|0|0|0" (padded)

Encrypted Text = b1 66 24 f1 bb 62 45 c8 4f
5f0 3c 7d 4

original : 5 M Jisan