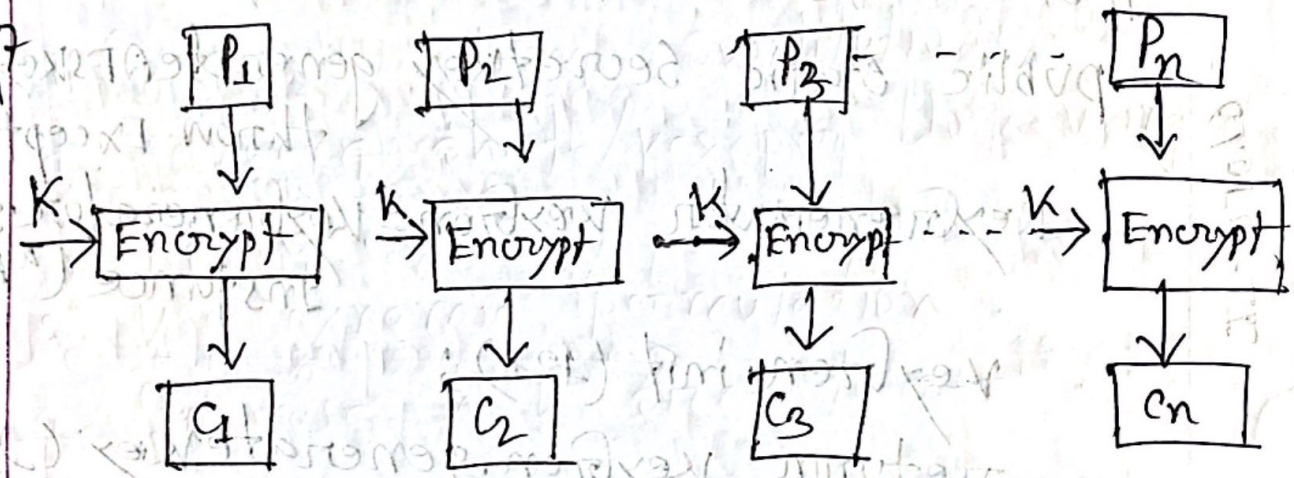
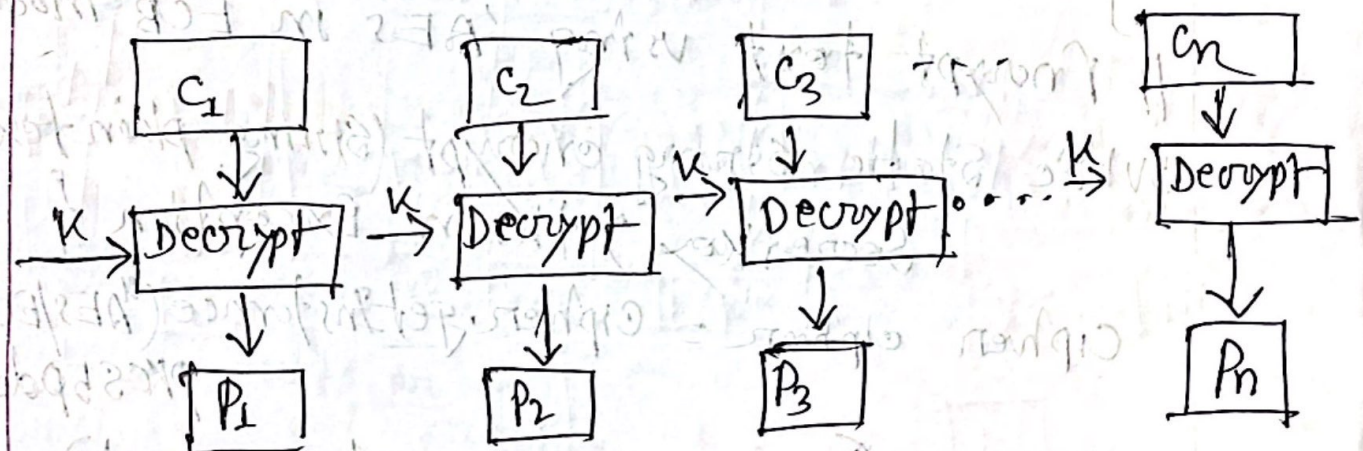
Assignment IT-21042

IT-21042

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

Block diagram:





IT-21042

Java implementation of ECB

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;
public class ECBModeExample {
// Generate a sample AES key (128 bit)
public static SecretKey generateAESKey() throws
                                    throw Exception {
    KeyGenerator keyGen = KeyGenerator.get
                                    Instance("AES");
    keyGen.init(128);
    return keyGen.generateKey();
}
// Encrypt text using 'AES in ECB mode$.
public static String encrypt(String plainText,
        SecretKey) throw Exception {
Cipher cipher = Cipher.getInstance("AES/ECB/
                                    PKCS5padding");
cipher.init(cipher.ENCRYPT_MODE, secret);
```

IT-21042

IT-2-10YR

```
byte[] encryptedBytes = cipher.doFinal(plainText.getBytes());
return Base64.getEncoder().encodeToString(encryptBytes);
}

// Decrypt cipher text using AES in ECB mode
public static String decrypt(String encryptedText, SecretKey secretKey) throws Exception {
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(cipher.DECRYPT_MODE, secretKey);
byte[] decryptedBytes = cipher.doFinal(Base64.getDecoder().decode(encryptedText));
return new String(decryptedBytes);
// Example
public static void main(String[] args){
try {
    secretKey = generateAESKey();
    String original = "Hello cyber world!";
    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);
```

IT-21042

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

Output:

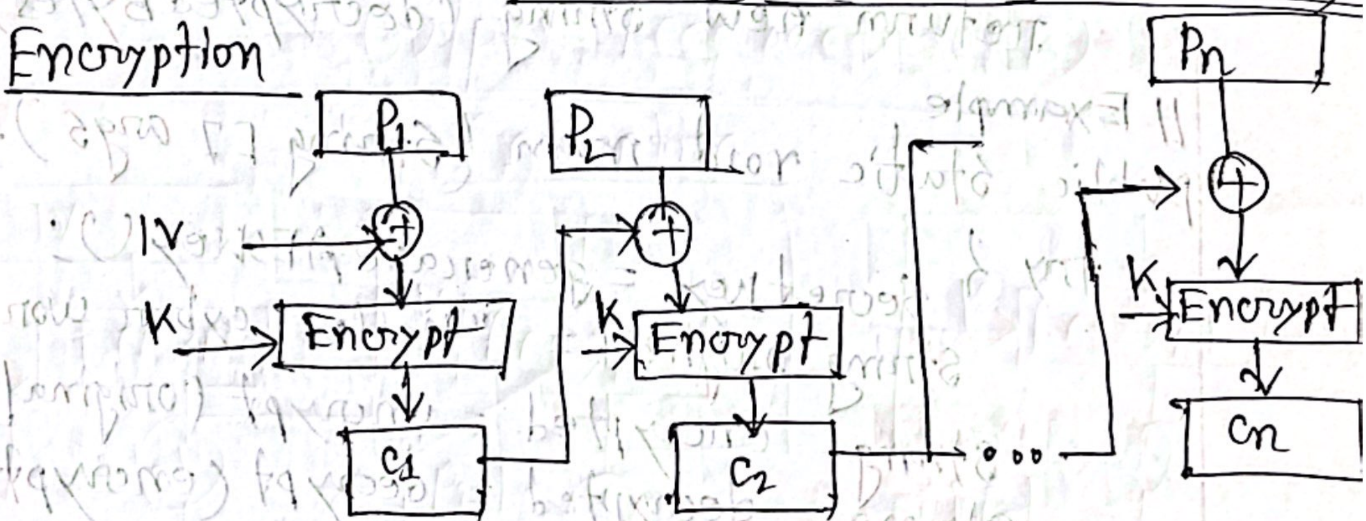
Original Text : Hello cyber world !

Encrypted Text : fRQ+NYKBP3A8cX1vq1gwf1an[illegible]g#

Decrypted Text : Hello cyber world !

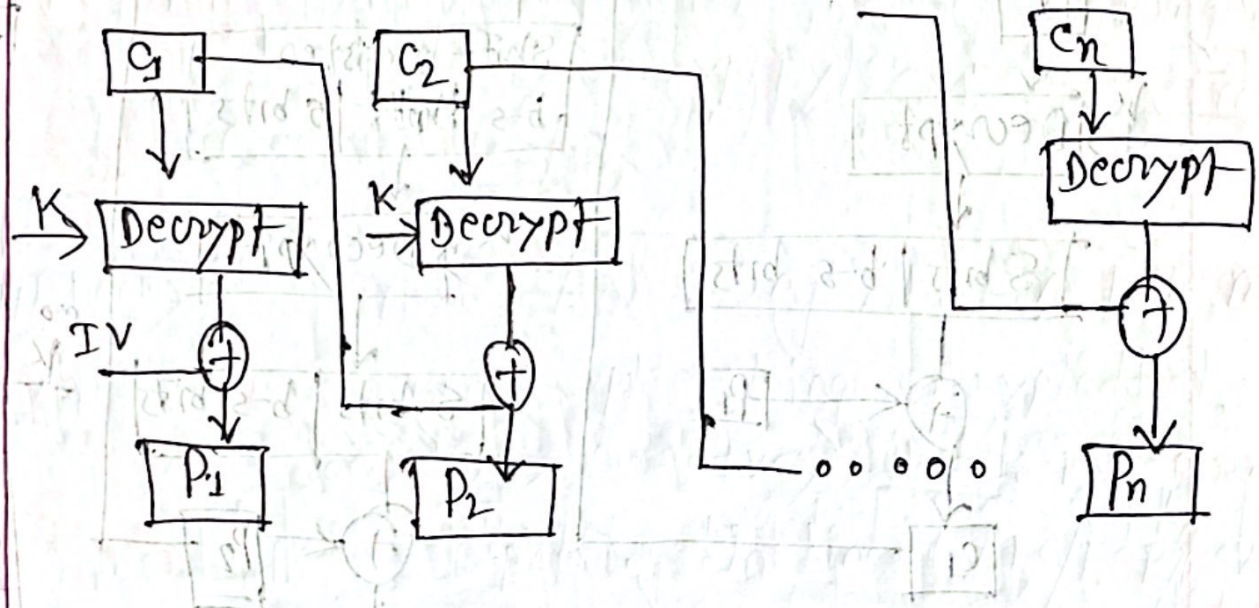
CBC (cipher Block chaining)

Encryption



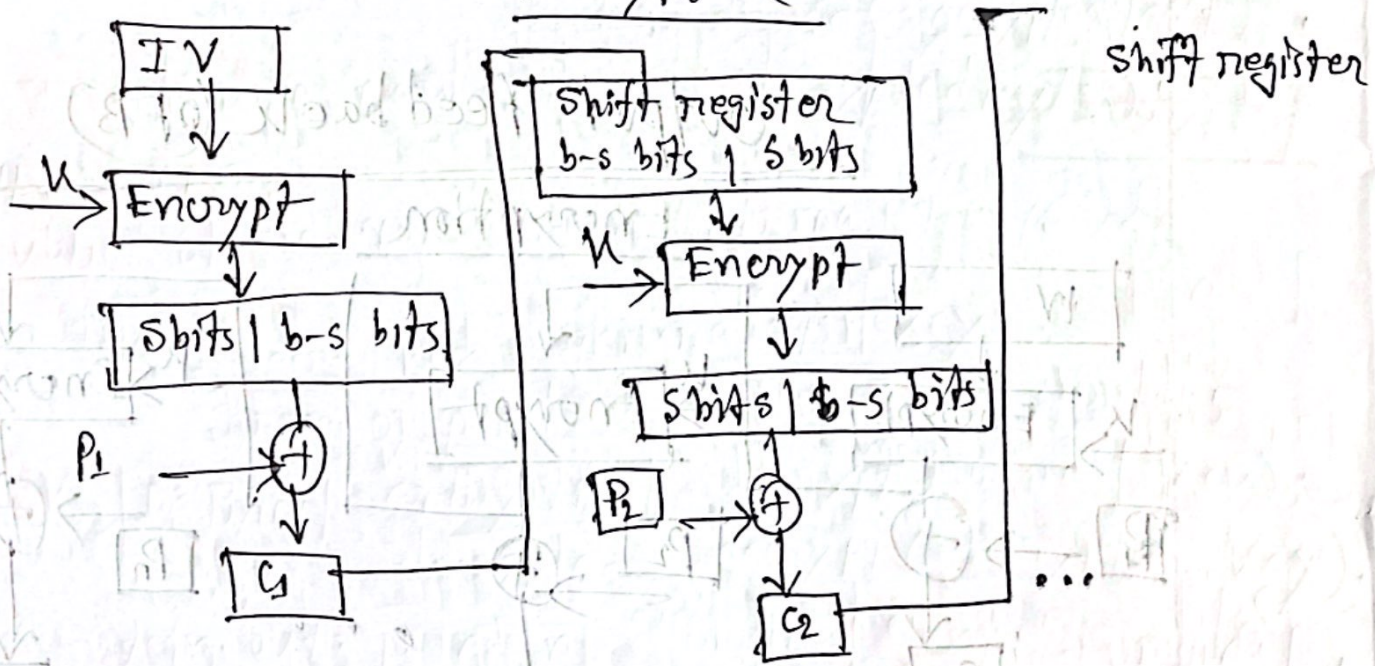
IT-21097

Decryption :



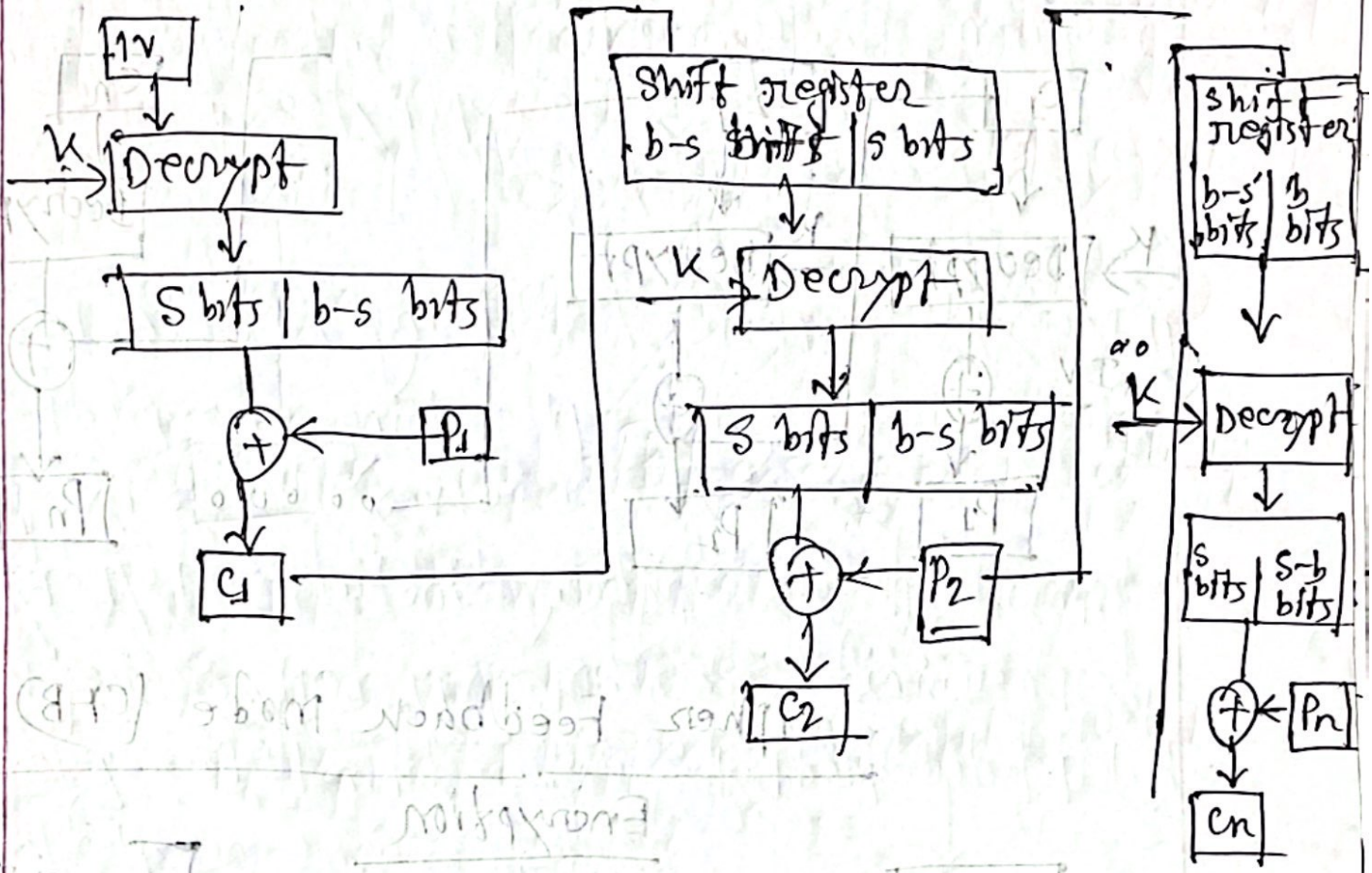
Cipher Feedback mode (CFB)

Encryption



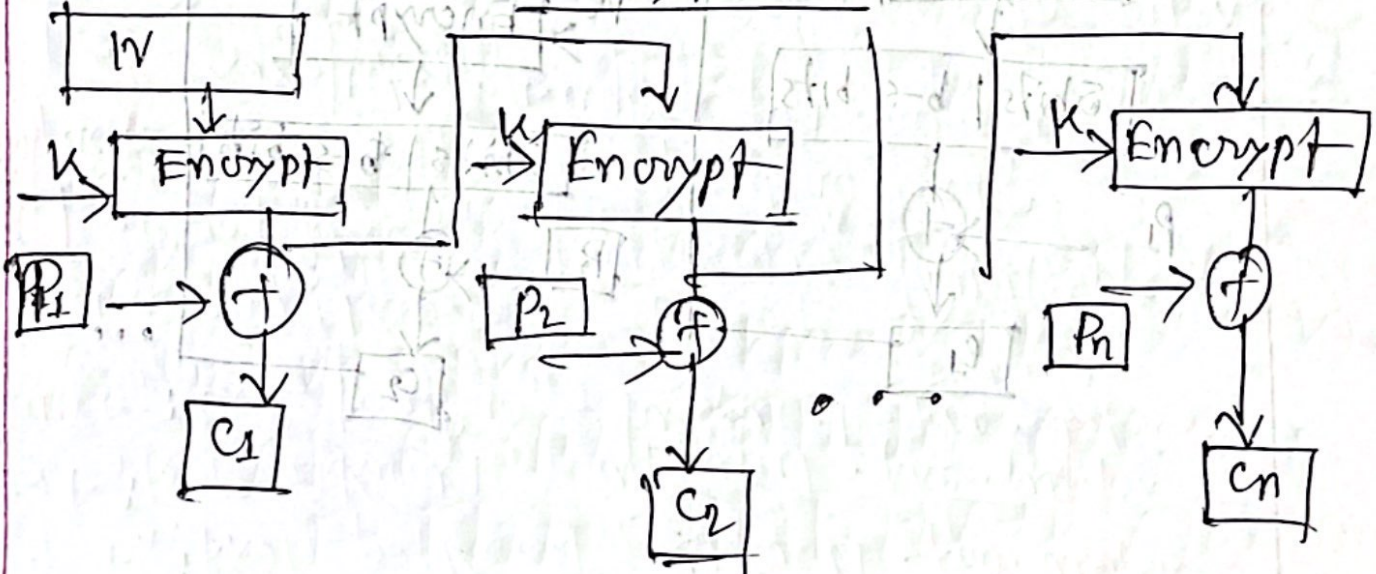
IT-2104/2

Decryption



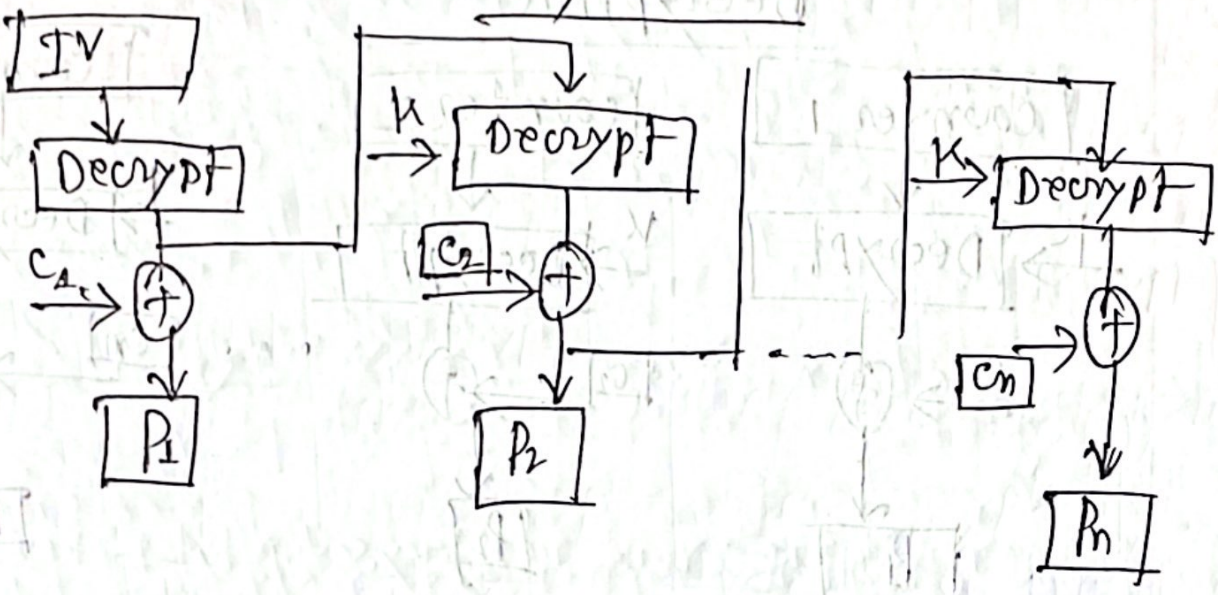
Output feedback (OFB)

Encryption



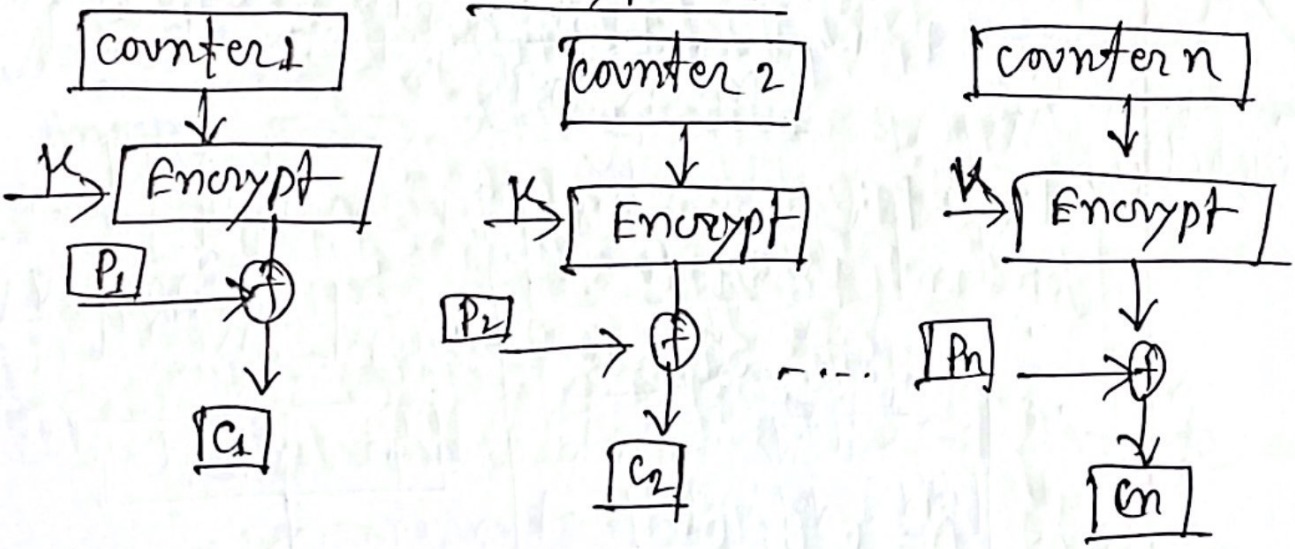
IT-21042

Decryption



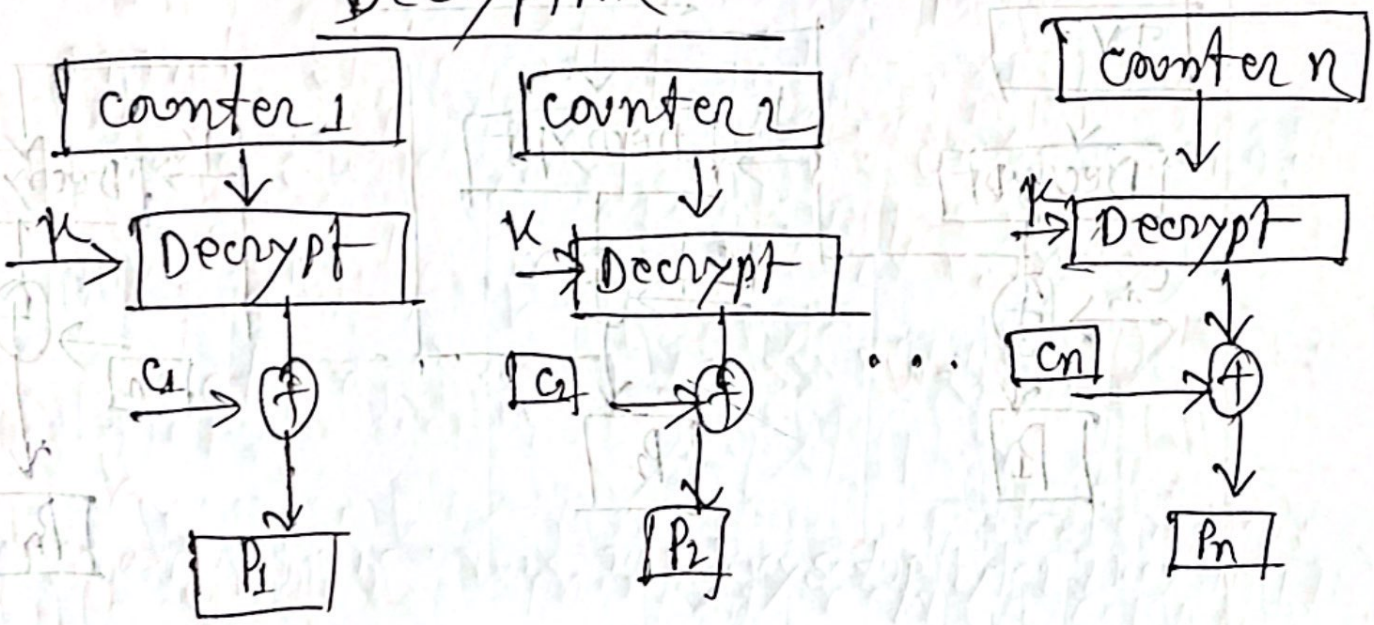
counter mode (CTR)

Encryption



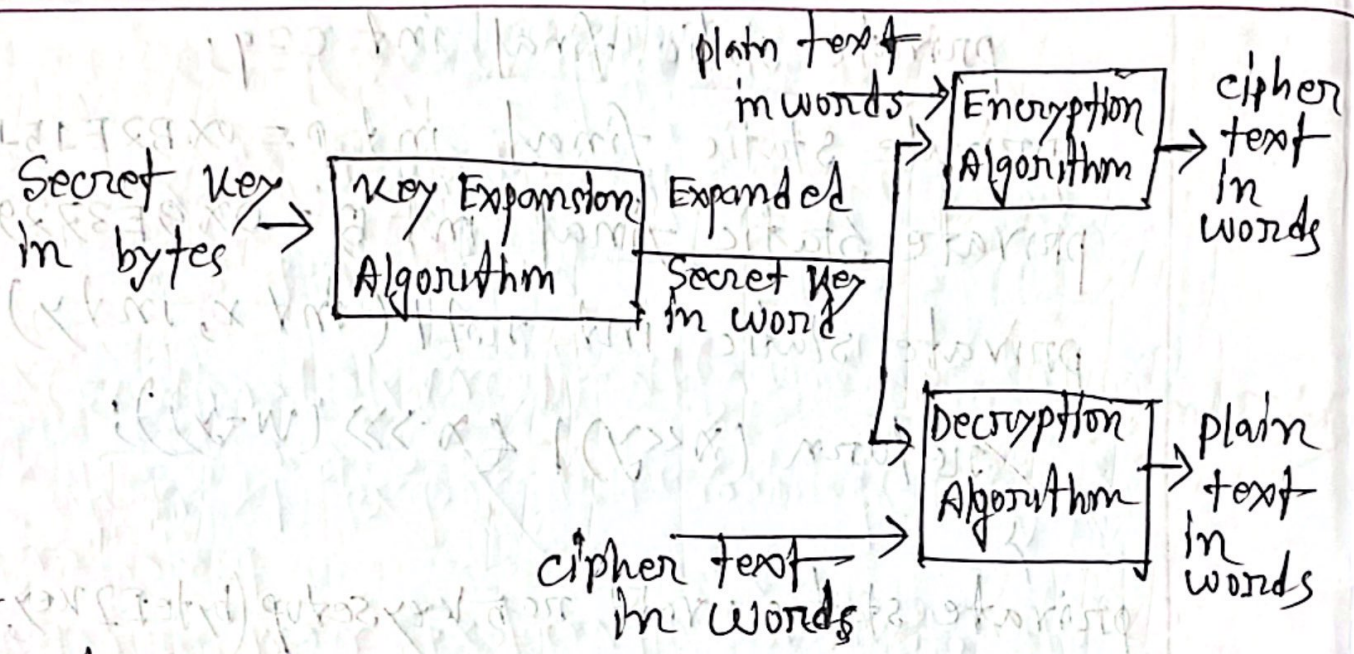
IT-21042

Decryption



IT-2-1042

RC5



IT-2-1042

code :

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

public class RC5FX1 extends Application {
    private static final int W = 32;
    private static final int R = 12;
    private static final int B = 16;
```

IT-21042

```
private static final int c=4;
private static final int p=0xB7E15163;
private static final int q=0x9E3779B9;
private static int rotl(int x, int y){
    return (x<<y)|(x>>>(w-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=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*Math.max(2*(r+1),c); ++k){
        A=S[i]=rotl(S[i]+A+B, 3);
        B=L[j]=rotl(L[j]+A+B,(A+B)%w);
```

IT-21042

```
        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 = ROTL(A^B, B) + S[2*i];
        B = ROTL(B^A, A) + S[2*i+1];
    }
    data[0] = A;
    data[1] = B;
}
private static String encryptText(String plainText){
    byte[] key = new byte[B];
    int[] S = new int[2*(R+1)];
```

```
    rc5.keySetup(key, 5);
    byte[] plainBytes = plainText.getBytes(StandardCharsets.UTF_8);
    int paddedLength = ((plainBytes.length + 7) / 8) * 8;
    byte[] padded = Arrays.copyOf(plainBytes, paddedLength);
    StringBuilder cipherText = new StringBuilder();
    ByteBuffer buffer = ByteBuffer.wrap(padded);
    while (buffer.hasRemaining()) {
        int A = buffer.getInt();
        int B = buffer.getInt();
        int[] data = {A, B};
        rc5.encrypt(5, data);
        cipherText.append(String.format("%08X%08X", data[0], data[1]));
    }
    return cipherText.toString().trim();
}
```

IT-21042

```
@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()){
        String cipherText=encryptText(plainText);
    }
    else{output.setText("please enter some text");
    }
});
    VBox layout = new VBox(10,input,encryptButton,
                           output);
    layout.setStyle("-fx-padding: 20;");
```

IT-21042

```
@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()){
    String cipherText=encryptText(plainText);
  }
  else{output.setText("please enter some text");
  }
});
  VBox layout = new VBox(10,input,encryptButton,
                         output);
  layout.setStyle("-fx-padding: 20;");
```

IT-21047

```
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: hello world

padding into 64 bit blocks

Block 1: "hello wo" (8 bytes)

Block 2: "rld\0\0\0\0" (padded)

Encrypted output:

07c9f2a2 b1552b56 24935838 5f02b8b3

original: Hello world.