Bindu

Block cipher mode of operation

Electronic code Book (ECB)

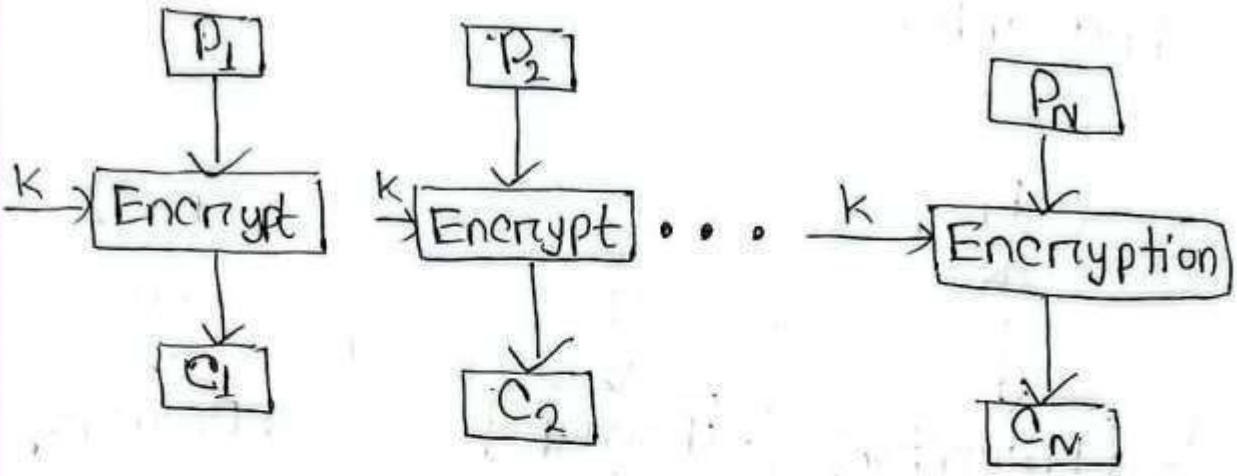


Fig: Encryption of ECB.

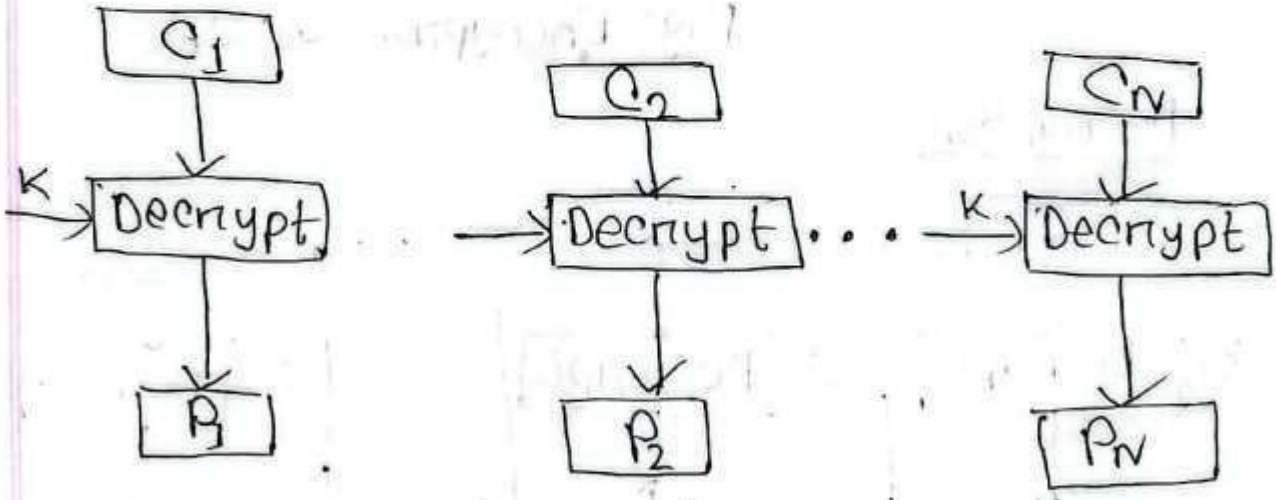


Fig:- Decryption of ECB.

Cipher Block Chaining

Encryption

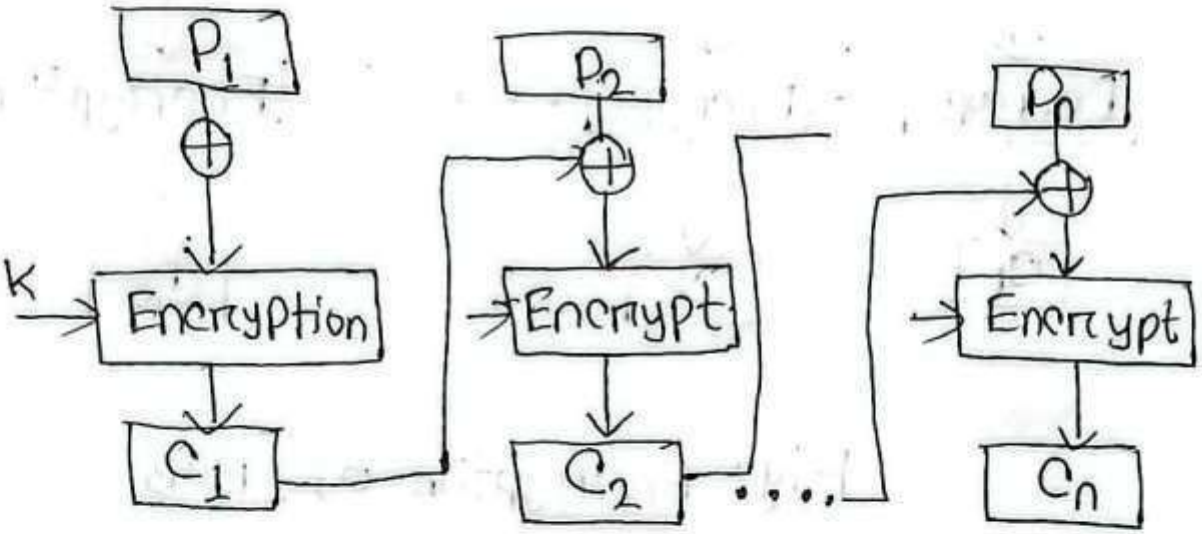


Fig:- Encryption of CBC.

Decryption

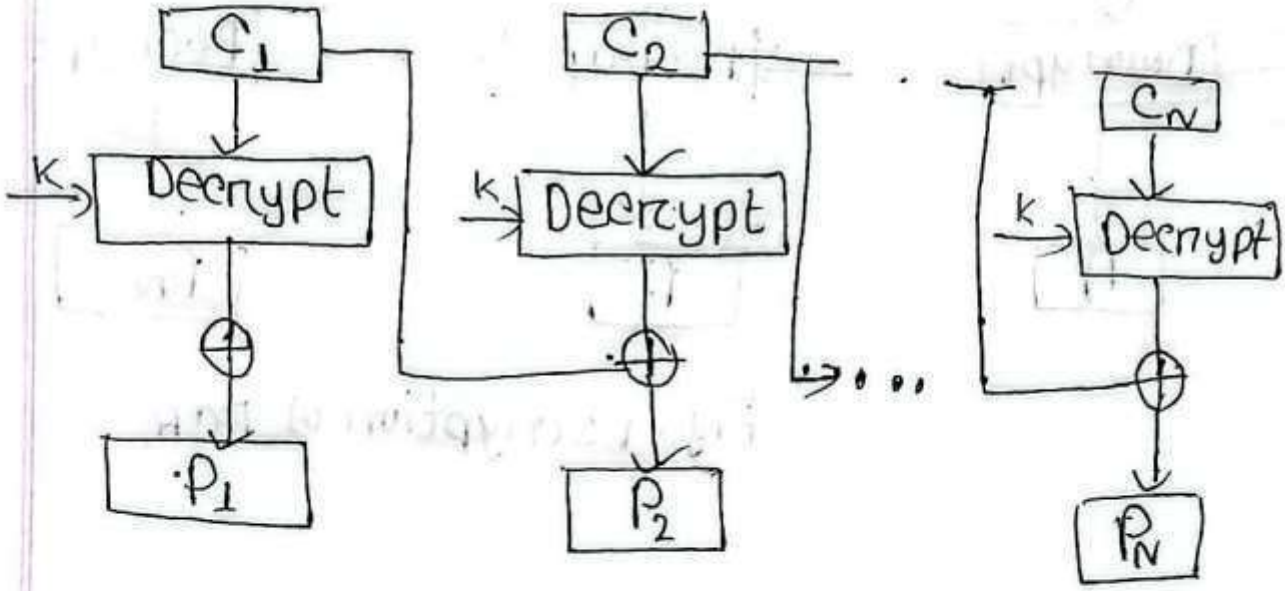


Fig: Decryption of CBC

Bindu

Cipher Feedback Mode(CFB)

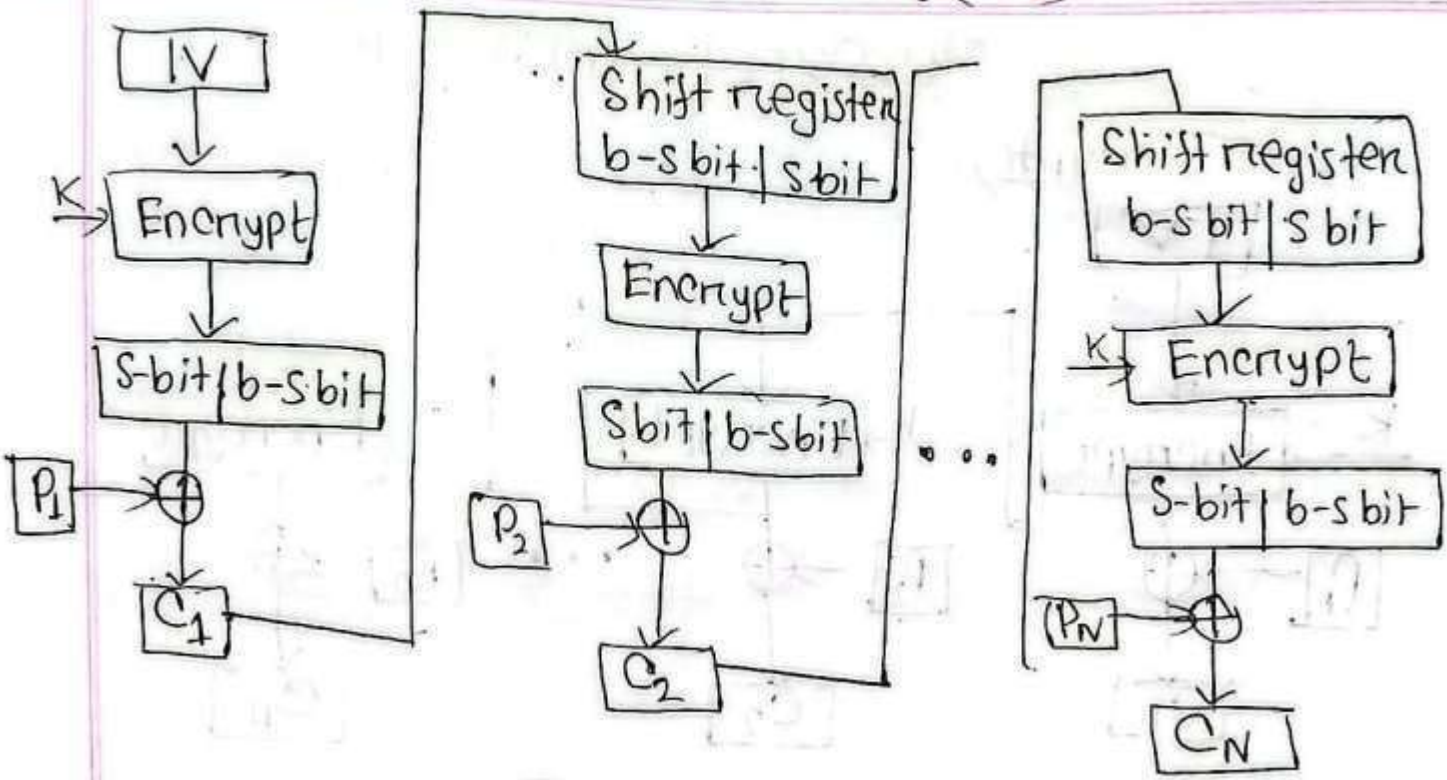


Fig:- Bto Encryption of CFB

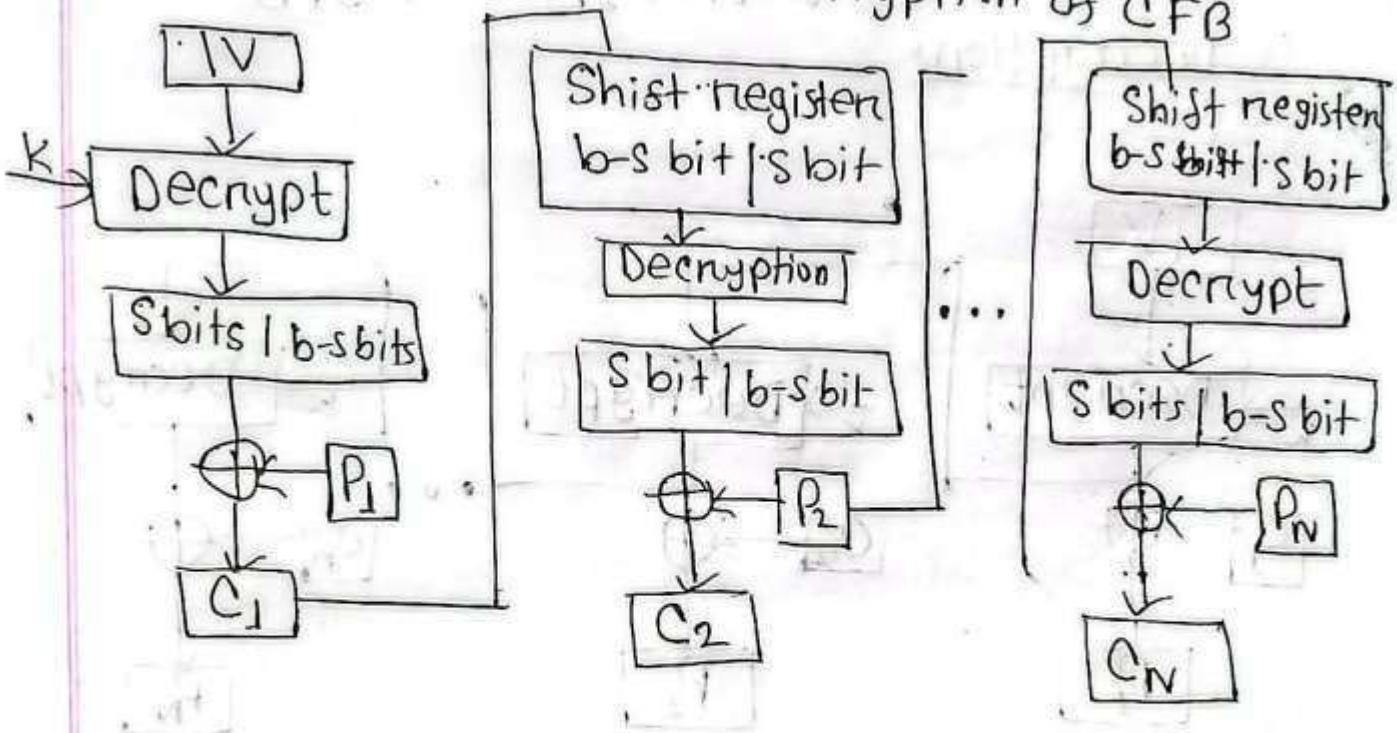


Fig: Decryption of CFB

Output FeedBack mode

Encryption

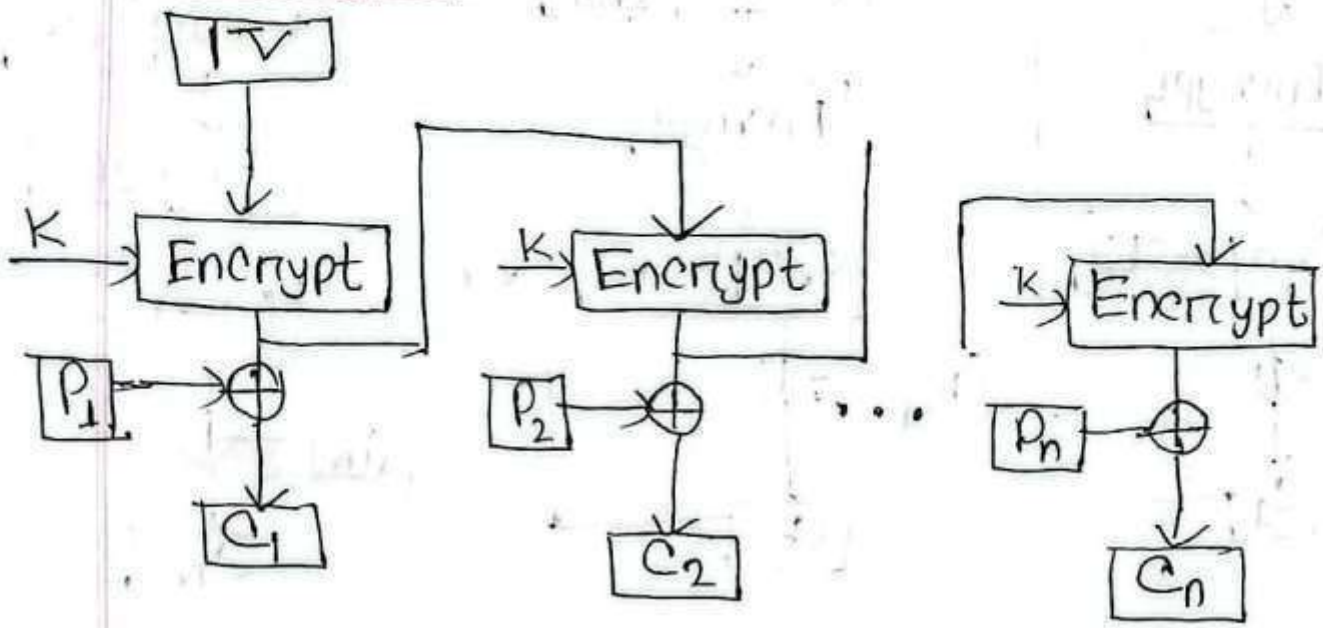


Fig:- Encryption of OFB

Decryption:-

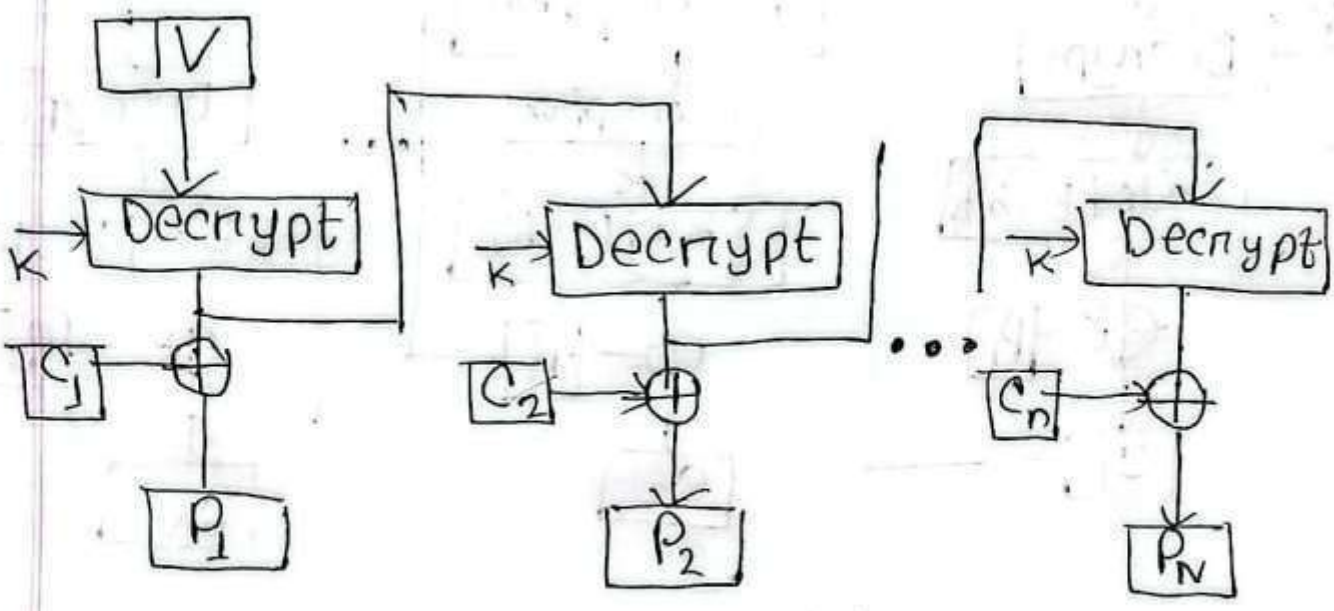


Fig:- Decryption of OFB.

Counter Mode

Encryption :-

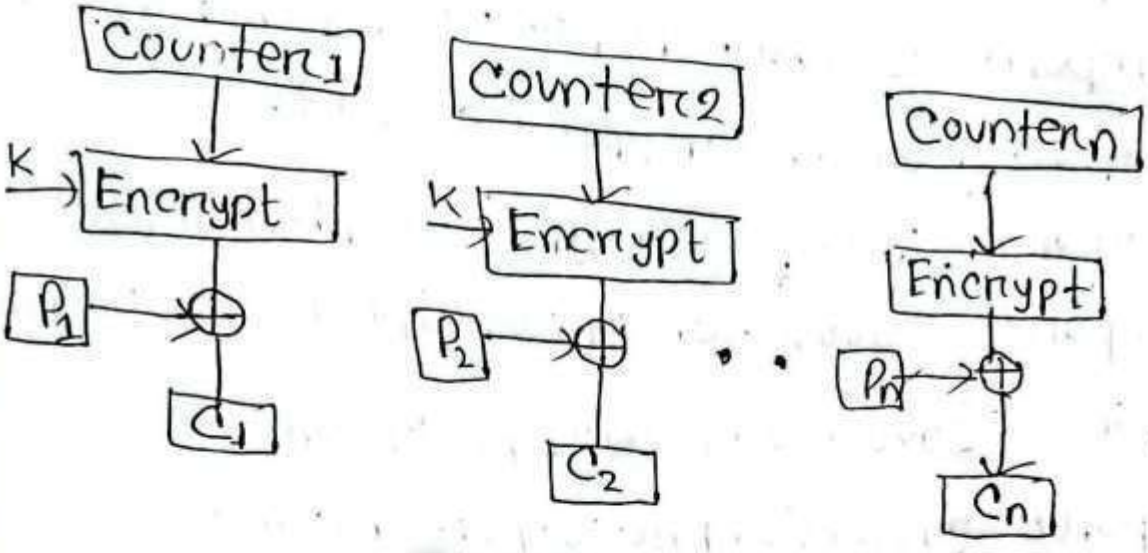


Fig:- Encryption of CM

Binder

Decryption

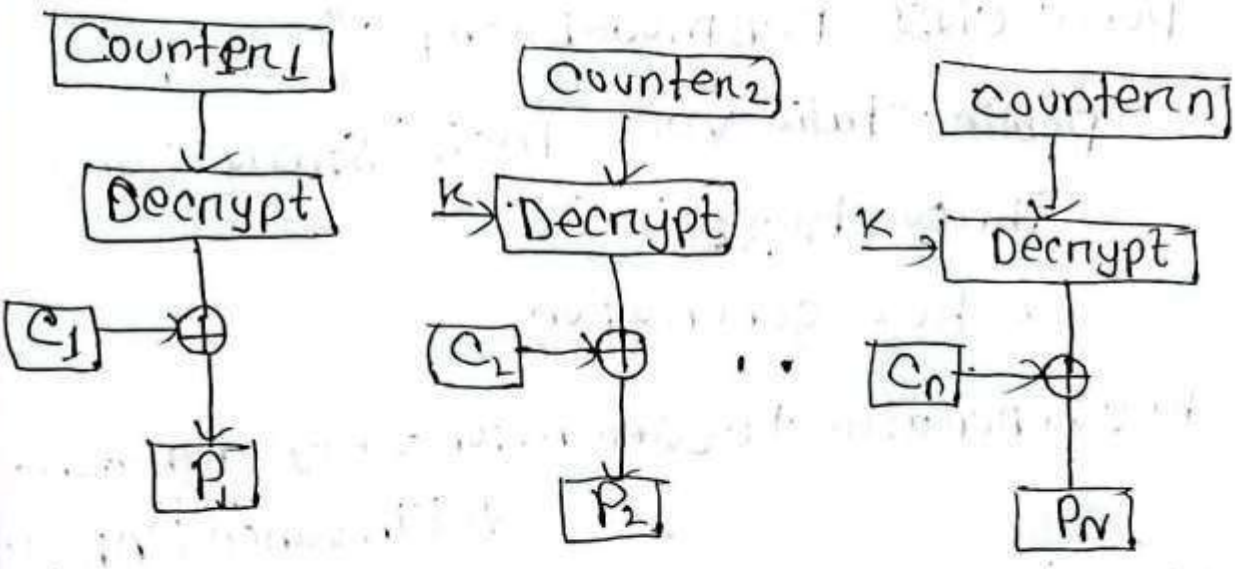


Fig: Decryption of CM

Code - Java For ECB

```
import Javax.crypto.Cipher;
import Javax.crypto.KeyGeneraton;
import Javax.crypto.Secretkey;
import Javax.crypto.CipherInputStream;
import Javax.crypto.CipherOutputSream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.Security.Key;

public class ECBModeExample{
    public static void main(String[]args)
        throw Exception{
        // key generation
KeyGeneraton keyGeneraton = keyGeneraton.
                            getInstance("AES");
Secretkey Secretkey = keyGeneraton.generatekey();
    // Cipher instance for ECB
Cipher cipher=Cipher.getInstance("AES/ECB/PKCS5padding");
```

Bindu

```
// Encryption of ECB mode
cipher.init(Cipher.ENCRIPT_MODE, secretkey);
byte[] encrypted = cipher.doFinal("This is a test".
                                  getBytes());
System.out.println("Encrypted: " + new String(encrypted));
// Decryption using ECB mode
Cipher.init(cipher.DECRYPT_MODE, secretkey);
byte[] decrypted = cipher.doFinal(encrypted);
System.out.println("Decrypted: " + new String(decrypted));
}
}
```

Bindu

Code for CBC

```
import javax.crypto.cipher;
import javax.crypto.keyGenerator;
import javax.crypto.Secretekey;
import javax.crypto.CipherInputStream;
import Javax.crypto.CipherOutputStream;
import Javax.crypto.spec.IvperameterSpec;
Public class CBCmode{
```

```
public Static void main (String[]args) throws Exception
{
KeyGeneraton keyGeneraton = KeyGeneraton.getInstance
                                ("AES");
Secretkey & Secretkey = keyGeneraton.getIenatekey();
  byte[]iv = new byte[16]
Ivpena ivpena = new Ippena(iv);
  // Ciphen instance for CBC                    Bindu
  Ciphen ciphen = Ciphen.getInstance(AES/CBC/
                                PKCSppadding");
//Encryption using CBC.
Ciphen.init(ciphen.ENCRYPT_MODE, Secretkey,
                                ivpena);
byte[.] encrypted = Ciphen.doFinal("This is me.
                                test".getBy
System.out.println(Encrypted(CBC); "+new
                                String(encrypted));
// Decryption
Ciphen.init(ciphen.DECRYPT_MODE, Secretekey,
                                ispena);
```

```
byte[] decrypted = cipher.doFinal(encrypted);
System.out.println("Decrypted (CBC): " + new String
                                        (decrypted));
}}
```

CFB mode code

Bindu

```
import javax.crypto.cipher;
import javax.crypto.keyGeneration;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class CFBmode{
public psvm(String[] arg) throws Exception{
keyGeneraton keyGeneraton = keyGeneraton.
                        getInstance("AES");
SecretKey SecretKey = keyGeneraton.generateKey();
byte[] iv = new byte[16];
IvParameterspec ivparameterspec = new IvparameterSpec(iv);
```

```
Cipher cipher = Cipher.getInstance("AES/CFB8/PKCS5Padding");

cipher.init(cipher.ENCRIPT_Mode, SecreteKey,
                    ivParametenSpace);

byte[] encrypted = cipher.doFinal("this is a text".
                    getBytes());

System.out.println(Encrypted(CFB):" + new
                    String(encrypted));

Cipher.init(Cipher.DECRYPT_MODE, secretkey,
                    ivparametenspec);

byte[] decrypted = cipherdoFinal(encrypted);
System.out.println("Decrypted(CFB):" + new
                    String(decrypted));
}
}
```

Binda

OFB Code.

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecreteKey;
import javax.crypto.spec.IvParameterSpec;
Public class OFBMode{
  PSVM{String[] arg) throws Exception {
KeyGenerator KeyGenerator = keyGenerator.
                    getInstance("AES");
SecreteKey SecreteKey = KeyGenerator.generateKey();
  byte[] IV = new byte[16];
  IvParameterSpec ivparameterspec = new IvParameterSpec
                                        (iv);
Cipher cipher = Cipher.getInstance("AES/OFB/PKCS5
                                    Padding");
```

Bindu

```
Ciphen.init(ciphen.ENCRIPT_MODE, Secretekey,
                    ivperametenspec);

byte[] encrypted = ciphen.doFinal("This is a text".
                    getByte());

System.out.println("Encrypted (OFB);"+new String
                    (encrypted));

Ciphen.inPt(ciphen.DECRIPT_MODE; Secretekey,
                    ivperametenspec);
byte[] decrypted = ciphen.doFinal(encrypted);
System.out.println("Decrypted (OFB);"+new
                    String(decrypted));
}
}
```

Bindu

Binder

Counter Mode

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class CTRMode{
    PSVM(String[]arg) throws Exception{
KeyGenerator keyGenerator = KeyGenerator.GetInstance
                                ("AES");
SecretKey secretkey= keyGenerator();
 byte[]iv = new byte(16);
IvParameterSpec ivparameterSpec = new IvParameter
                                    Spec(IV)
Cipher cipher =Cipher.getInstance("AES/CTR/NoPadding");
Cipher init(cipher.ENCRYPT_mode, secretkey,
                                ivparamet-erSpec);
byteT]encrypted = cipher.doFinal("This is a test.")
                                getBytes());
```

```
System.out.println("Encrypted(CTR):" + new
                    String(encrypted));
Ciphen.init(Ciphen.DECRYPT_mode, Secnetkey
                    ivperametenspec);
byte[] decrypted = Ciphen.doFinal(encrypted);
System.out.println("Decreypted(CTR):" new
                    String.(decrypted));
}
)
```

Bindu

Output

```
CBC
Encrypted (CBC): ·??h??4~?Hr?_?
Decrypted (CBC): This is a test.

=== Code Execution Successful ===
```

Output

```
CFB
Encrypted (CFB): ·?N?????·?dp??·
Decrypted (CFB): This is a test.

=== Code Execution Successful ===
```

Output

```
CTR
Encrypted (CTR): ·?Q??(?Bm??&~??
Decrypted (CTR): This is a test.

=== Code Execution Successful ===
```

Output

```
ECB
Encrypted: ??6·??JD?vx?·??
Decrypted: This is a test.

=== Code Execution Successful ===
```

Output

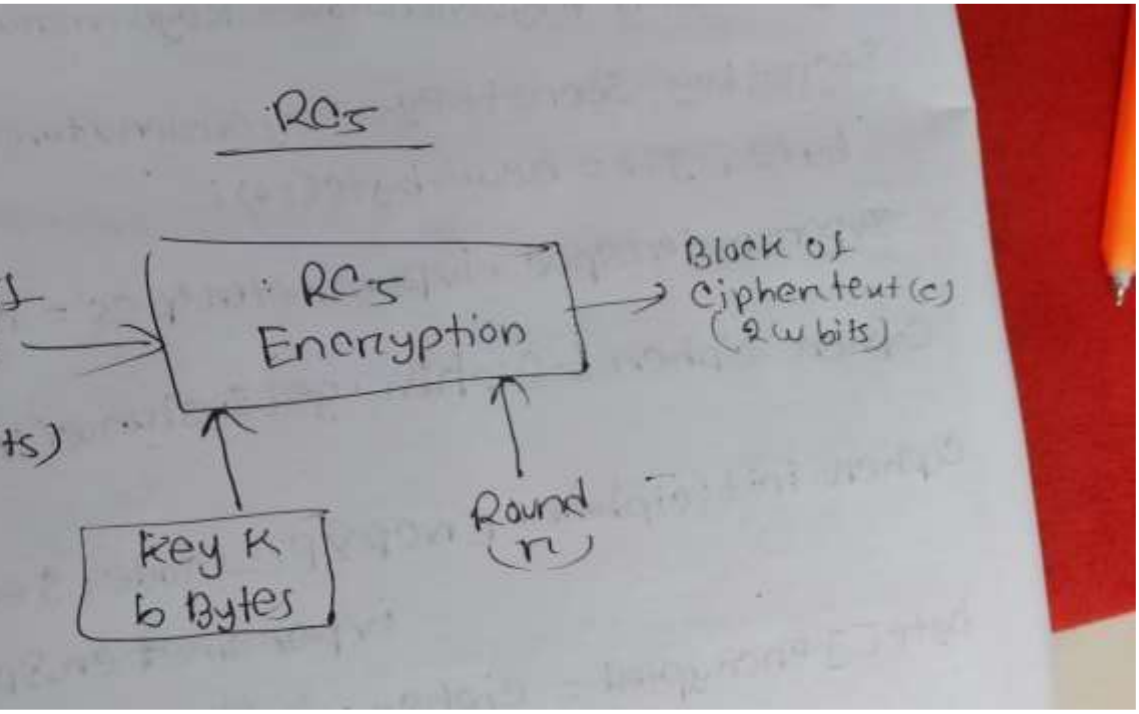
```
OFB
Encrypted (OFB): ??·?P?.a·?·
%·?
Decrypted (OFB): This is a test.

=== Code Execution Successful ===
```

```
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;     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);
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)];
      rc5KeySetup(key, S);

      byte[] plainBytes = plainText.getBytes(StandardCharsets.UTF_8);        int paddedLength = ((plainBytes.length + 7) / 8) * 8;
      byte[] padded = Arrays.copyOf(plainBytes, paddedLength);

      StringBuilder cipherHex = new StringBuilder();
      ByteBuffer buffer = ByteBuffer.wrap(padded);

      while (buffer.hasRemaining()) {
int A = buffer.getInt();            int B =
buffer.getInt();            int[] data =
{A, B};            rc5Encrypt(S, data);
         cipherHex.append(String.format("%08x %08x ", data[0], data[1]));
      }

      return cipherHex.toString().trim();
   }

   @Override
   public void start(Stage stage) {
TextField input = new TextField();
      input.setPromptText("Enter English words 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);
            output.setText("Cipher Text:\n" + cipherText);
         } else {
            output.setText("Please enter some text.");
         }
      });

      VBox layout = new VBox(10, input, encryptButton, 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);
    }
}
```



RC5

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import java.security.Security;
import java.util.Base64;
public class RC5 {
    public static void main(String[] args) throws Exception {
    Security.addProvider(new BouncyCastleProvider());
    KeyGenerator keyGen = KeyGenerator.getInstance("RC5", "BC");
    keyGen.init(128);
    SecretKey key = keyGen.generateKey();
    Cipher cipher = Cipher.getInstance("RC5/ECB/PKCS7Padding", "BC");
    String plainText = "Hello RC5";
    System.out.println("Plain: " + plainText);
    cipher.init(Cipher.ENCRYPT_MODE, key);
    System.out.println("Encrypted (Base64): " + Base64.getEncoder().encodeToString(encrypted));
    cipher.init(Cipher.DECRYPT_MODE, key);
    byte[] decrypted = cipher.doFinal(encrypted);
    System.out.println("Decrypted: " + new String(decrypted));
    }
}
```

Bindu