Bindu

Block cipher mode of operation

Electronic code Book (ECB)

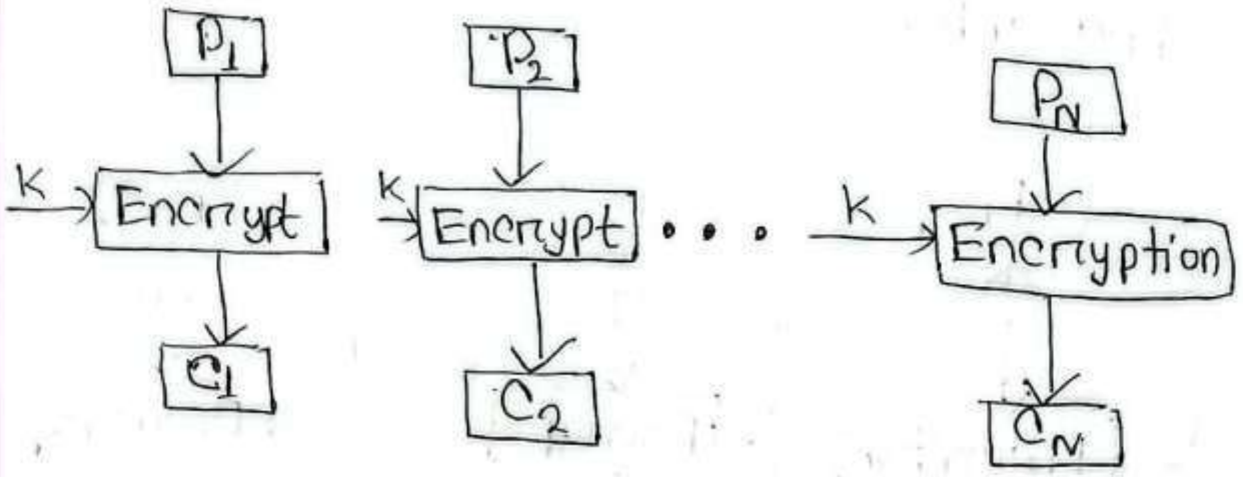


Fig: Encryption of ECB.

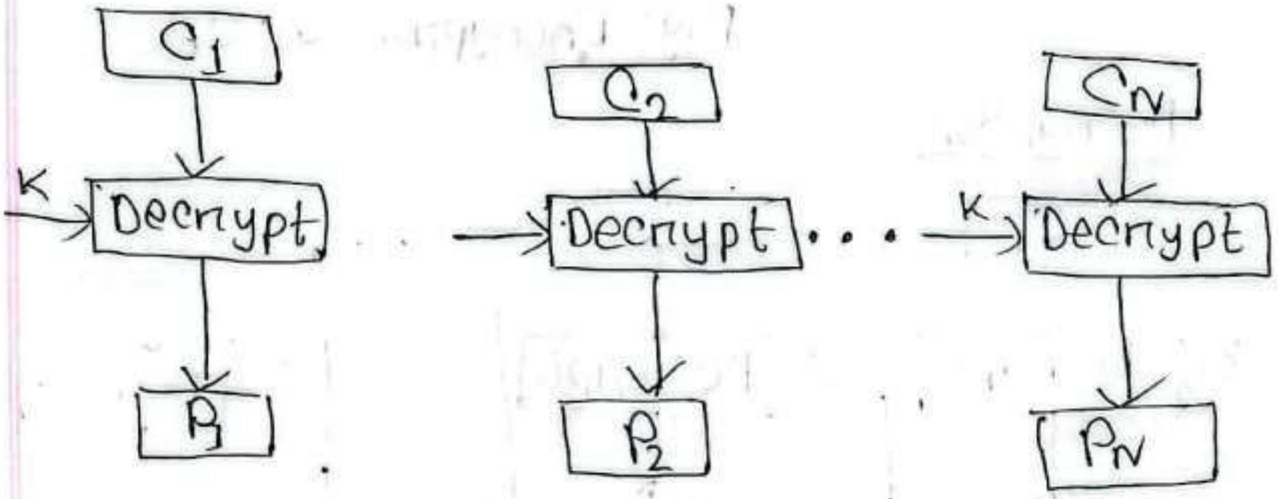


Fig:- Decryption of ECB.

Binda

Cipher Block Chaining

Encryption

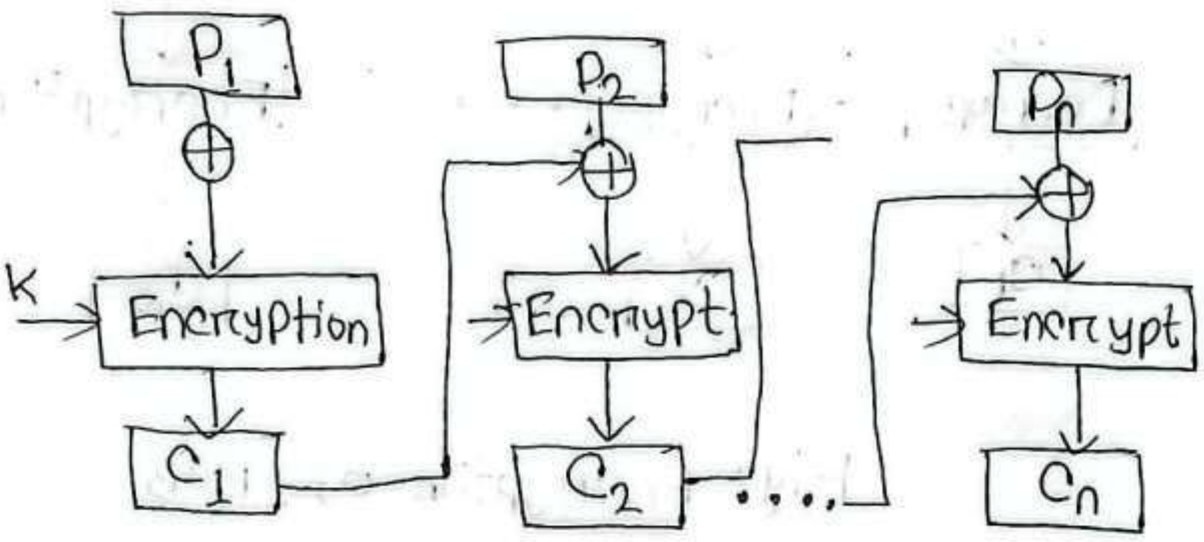


Fig:- Encryption of CBC.

Decryption

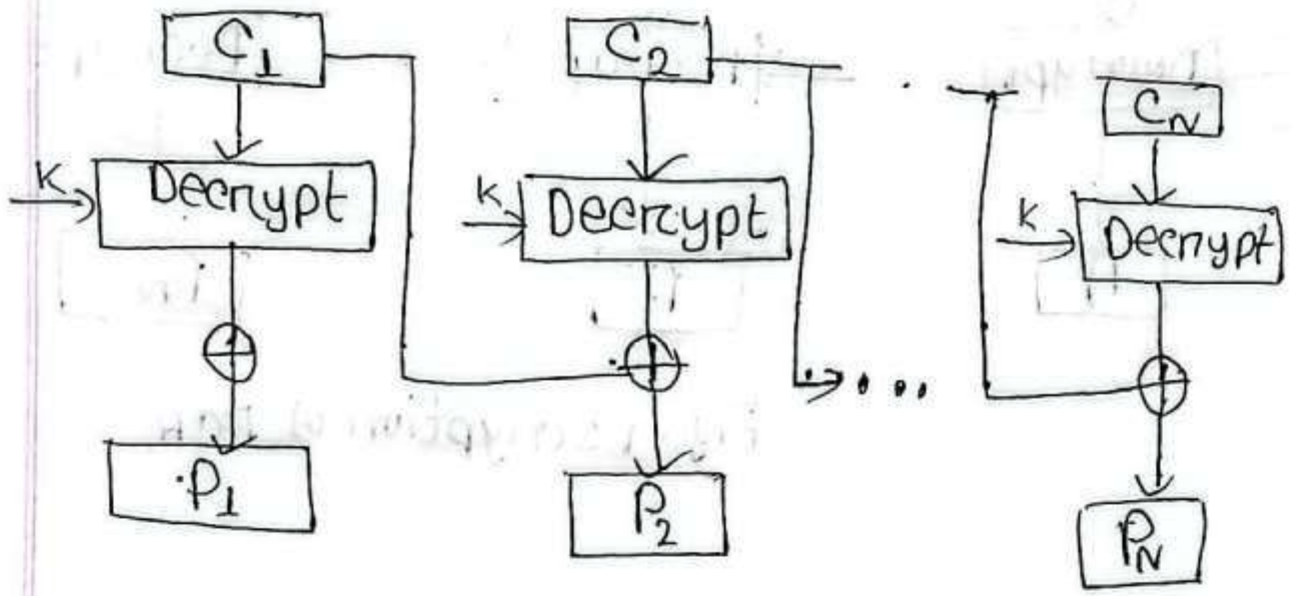


Fig: Decryption of CBC

Bindu

Cipher Feedback Mode(CFB)

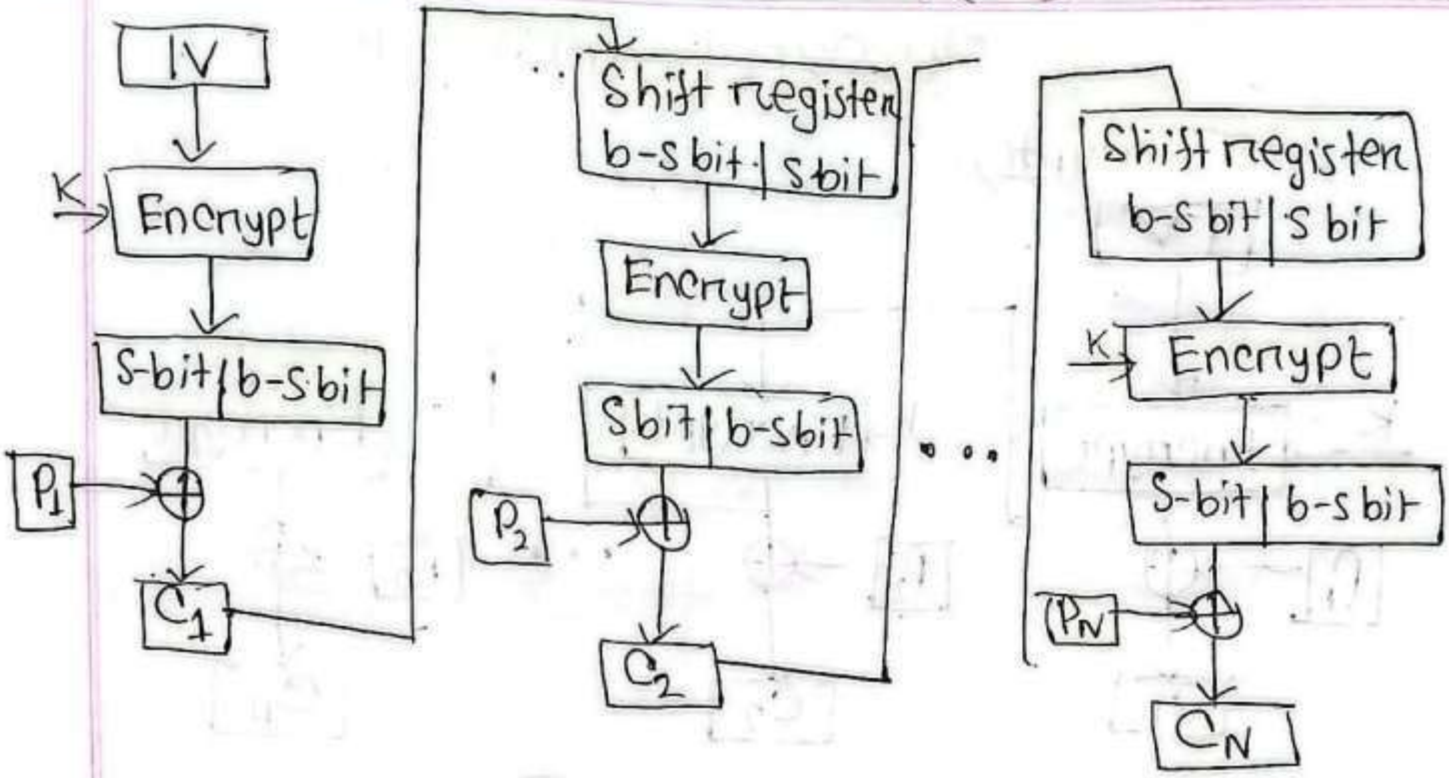


Fig:- ~~Bto~~ Encryption of CFB

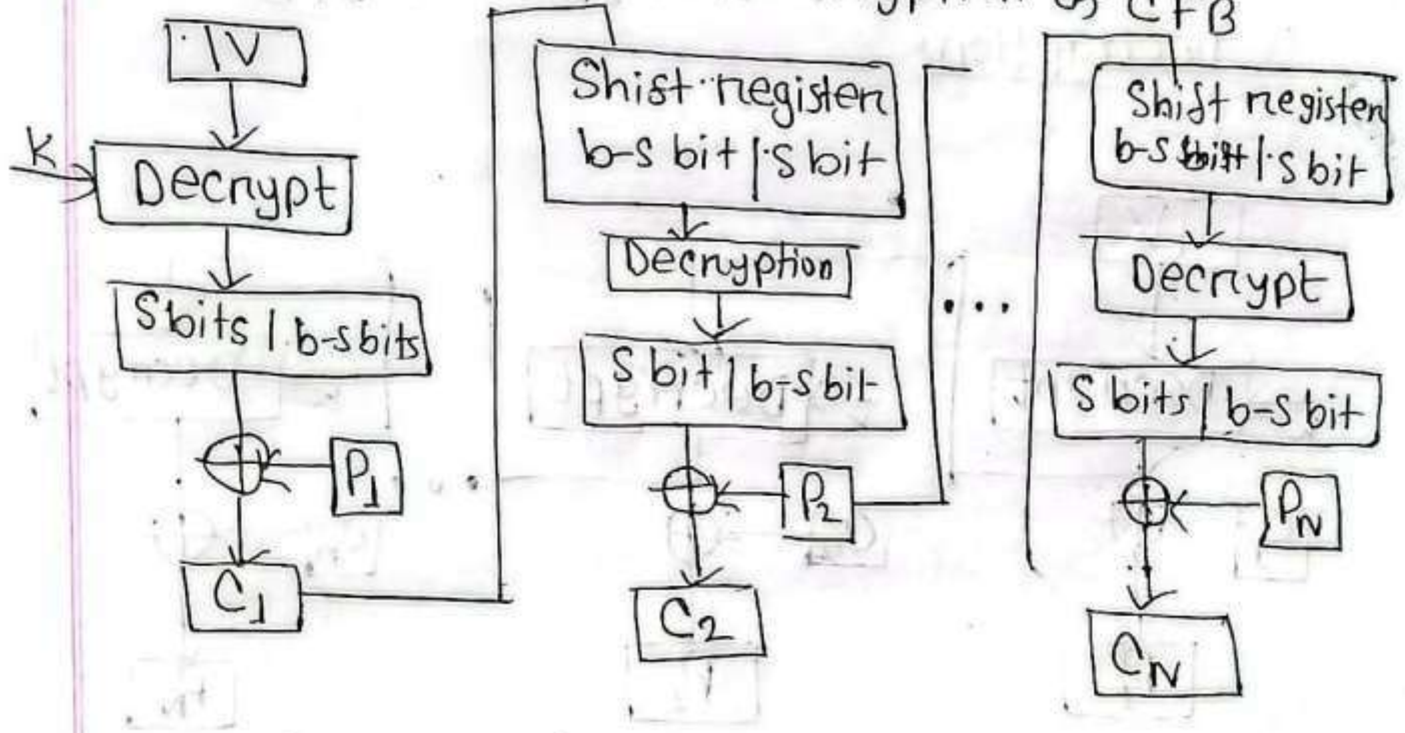


Fig: Decryption of CFB

Output FeedBack mode

Encryption

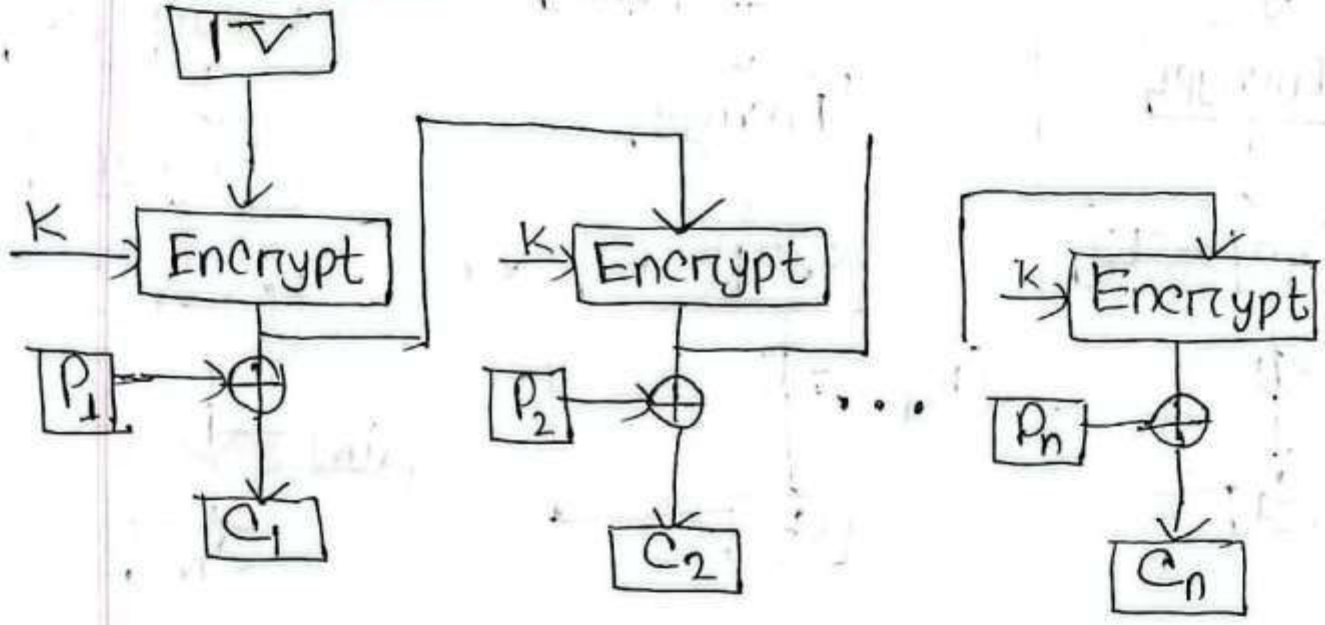


Fig: Encryption of OFB

Decryption:-

Bindu

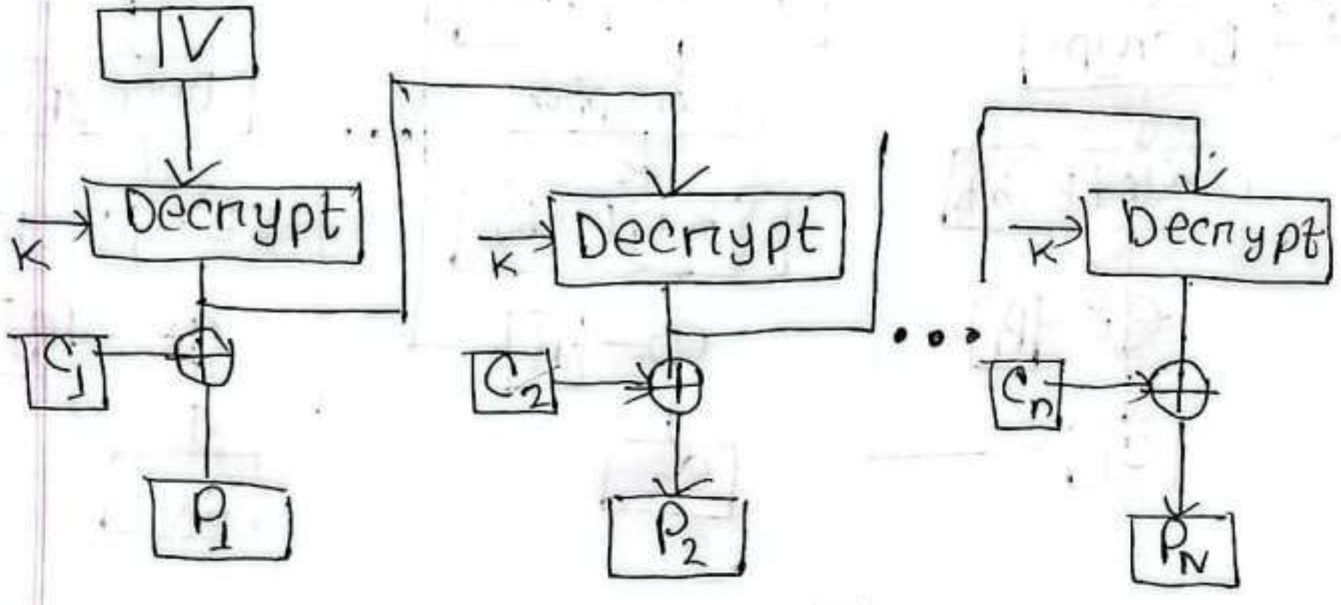


Fig:- Decryption of OFB.

CamScanner

Counter Mode

Encryption :-

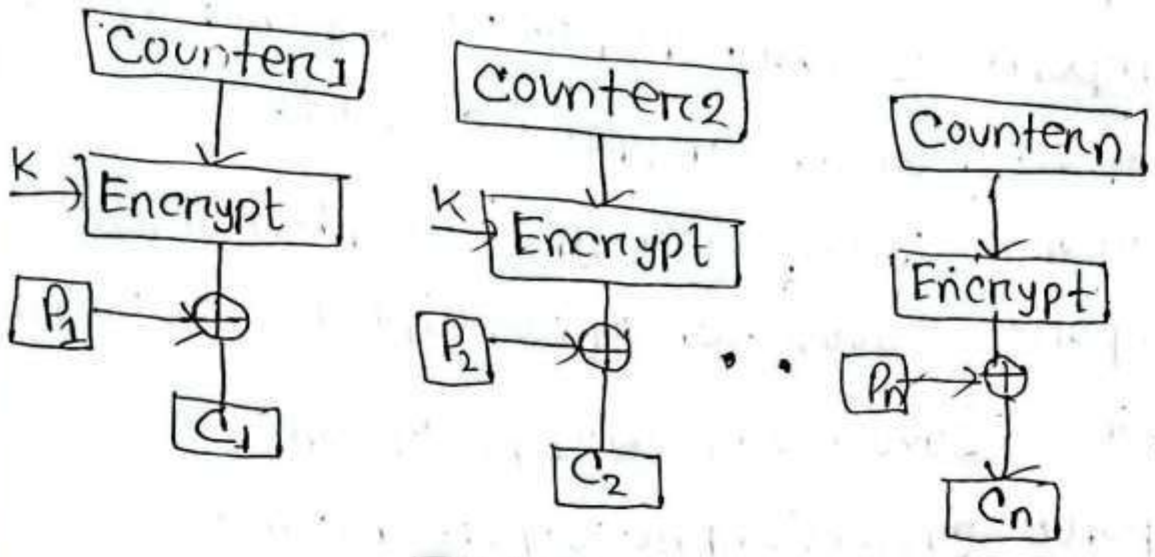


Fig:- Encryption of CM

Bindu

Decryption

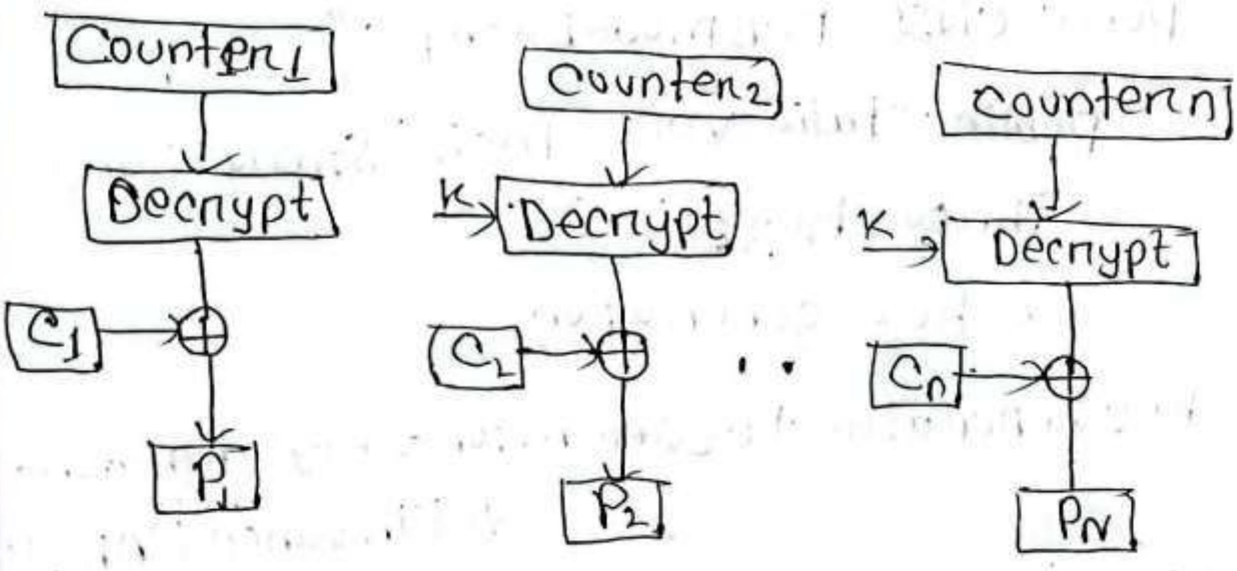


Fig: Decryption of CM

Code-Java-For ECB

```
import Javax.crypto.Ciphen;
import Javax.crypto.KeyGeneraton;
import Javax.crypto.Secretkey;
import Javax.crypto.CiphenInputStream;
import Javax.crypto.CiphenOutputSneam;
import Java.io.FileInputStream;
import Java.io.FileOutputStream;
import Java.Security.key;

public class ECBModeExample{
    public Static void main(String []angs)
      throw Exception{
        // key generation
 keyGeneraton keyGeneraton = keyGeneraton.
                         getInstance("AES");
Secretkey Secretkey = keyGeneraton.generatekey();
    // ciphen instance fon ECB
Ciphen ciphen=Ciphen.getInstance("AES/ECB/PKCS5padding");
```

Bindu

CamScanner

```
// 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.Ivparameterspec;
Public class CBCmode{
```

```
public Static void main (String[]args) throws Exception
{
KeyGenerator keyGenerator = keyGenerator.getInstance
                                              ("AES");
Secretkey secretkey= keyGenerator.generatekey();
  byte[]iv =new byte[16]
IvPera ivpera = new IvPera(iv);
 // Cipher instance for CBC
 Cipher cipher = Cipher.getInstance (AES/CBC/
                                     PKCSPadding");
//Encryption using CBC.
Cipher.init(cipher.ENCRYPT_MODE, Secretkey,
                          ivpera);
byte[] encrypted = Cipher.doFinal ("This is me.
                                      test".getBytes
System.out.println (Encrypted(CBC) : "+ new
                   String(encrypted));
// Decryption
cipher.init(cipher.DECRYPT_MODE, Secretekey,
                  ispera);
```
Bindu

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

CFB Mode code

Bindu

```
import javax.crypto.ciphen;
import javax.crypto.keyGeneration;
import javax. crypto - secretekey;
import javax.crypto.space.Ivperameterspec;
public class CFBmode{
~~public~~ psvm(String[]args) throws Exception{
keyGeneraton keyGeneraton= keyGeneraton.
                        getInstance("AES");
Secretkey Secretkey = keyGeneraton.generatekey();
byte[] iv= new byte[16];
Ivperameterspec ivperameterspec= new Ivperameterspec
                                              (iv);
Ciphen ciphen = Ciphen . getInstance(AES/CFB8/PKCS5padding);

Ciphen. init (ciphen.ENCRIPT_Mode, Secnetekey,
                        ivpenametenSpace);

byte[] encnypted = ciphen . doFinal ("this is a test".
                                          getBytes());

System . out . println (Encrypted (CFB):"+ new
                         Stning (encnypted));

Ciphen . init(Ciphen . DECRYPT_MODE, secnetkey,
                        ivpanametenspec);

byte[] decnypted = ciphen.doFinal (encnypted);
System. out . println ("Decnypted (CFB):"+ new
                  Stning(decnypted));
}
}
```

Bindu

CamScanner

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 ,Secnetekey,
                        ivpenametenspec);

byte[] encnypted = ciphen.doFinal ("This is a test".
                                        getbyte());

System.out.println("Encnypted (OFB):"+~~Stew~~ new String
                                     (encnypted));

ciphen.inPt(ciphen.DECRIPT_MODE , Secnetekey,
                                    ivpenametenspec);
byte[] decnypted = ciphen.doFinal (encnypted);
System.out.println("Decnypted (OFB):"+new
                     String (decnypted));
  }
}
```

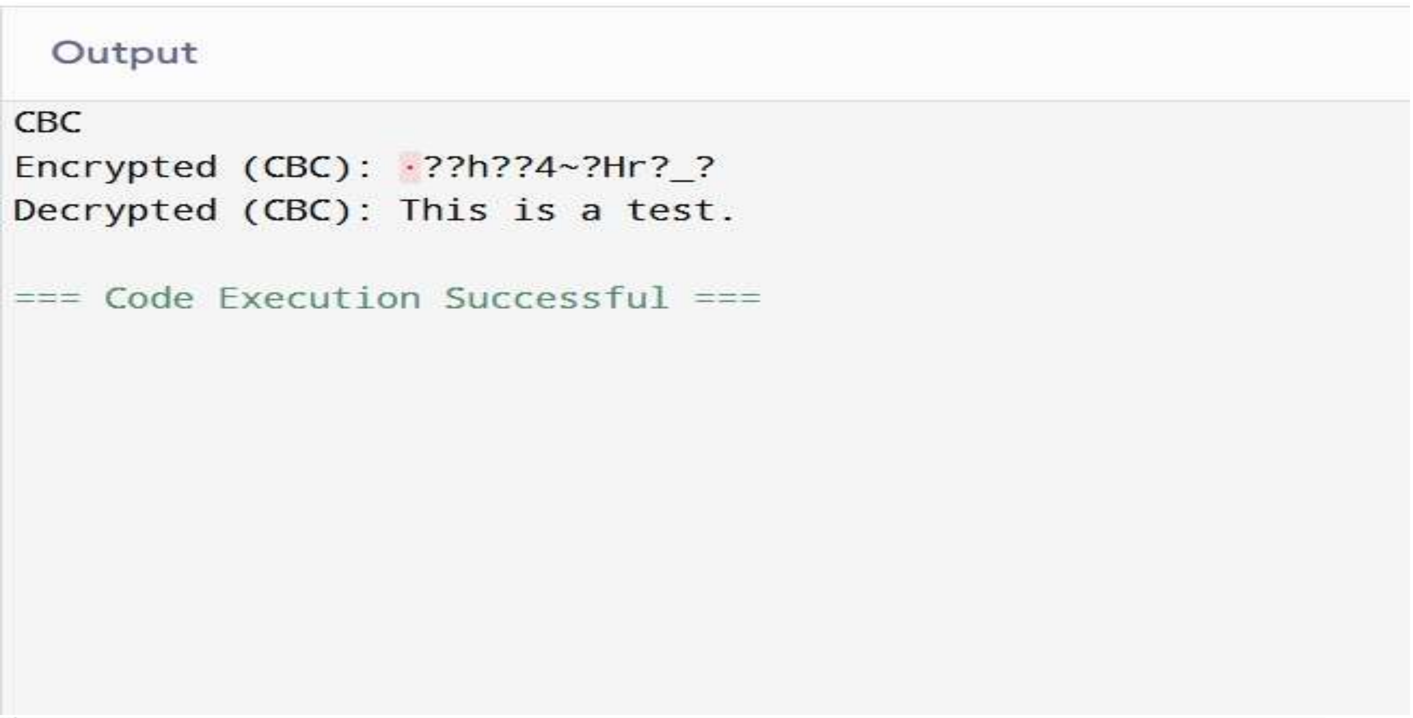
Bindu

Bindu

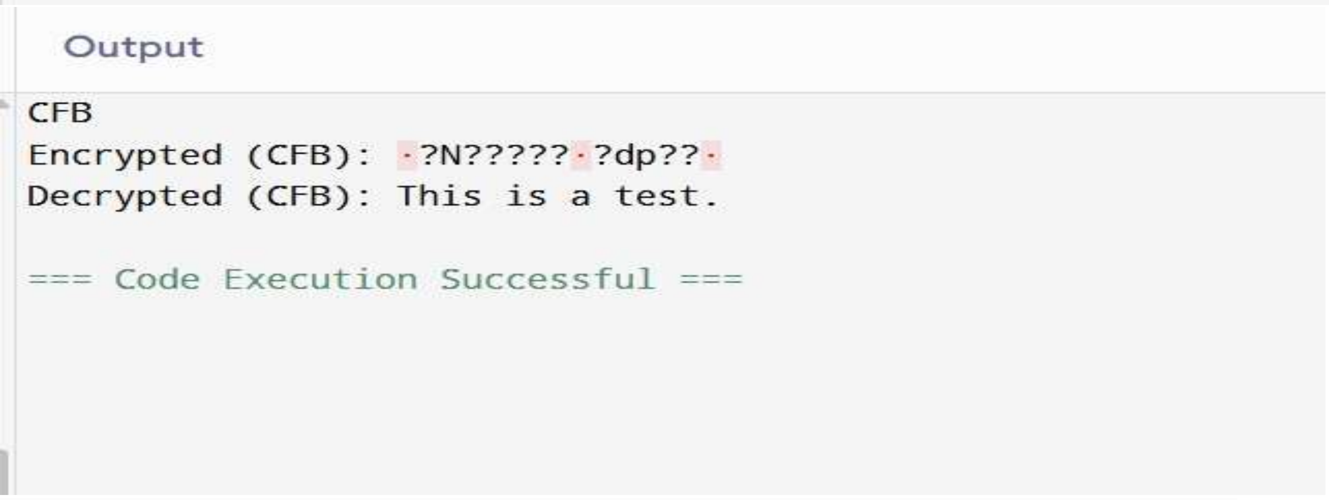
Counter Mode

```
import Javax.crypto.Cipher;
import Javax.crypto.KeyGenerator;
import Javax.crypto.SecretKey;
import Javax.crypto.spec.IvParameterSpec;
public class CTRMode{
    PSVM(String [Jarg] throws Exception{
KeyGenerator KeyGenerator = KeyGenerator.GetInstance
                                        ("AES");
SecretKey SecretKey= KeyGenerator();
 byte[]iv=new byte(16);
IvParameterSpec ,ivParameterSpec=new IpParameter
                                        Spec(IV)
Cipher cipher =Cipher.getInstance("AES/CTR/Nopadding")
Cipher init(cipher.ENCRYPT_mode, Secretkey,
                         ivParameterSpec);
byte[]encrypted = cipher.doFinal("This is a text."
                         getBytes());
System.out.println("Encrypted(CTR):"+new
                    String(encrypted));
Ciphen.init(Ciphen.DECRYPT_mode,Secnetkey
                    ivpenametenspec);
byte[]decnypted=Ciphen.doFinal(encnypted.)
System.out.println("Decneypted(CTR):"new
}                   String.(decnypted));
}
```

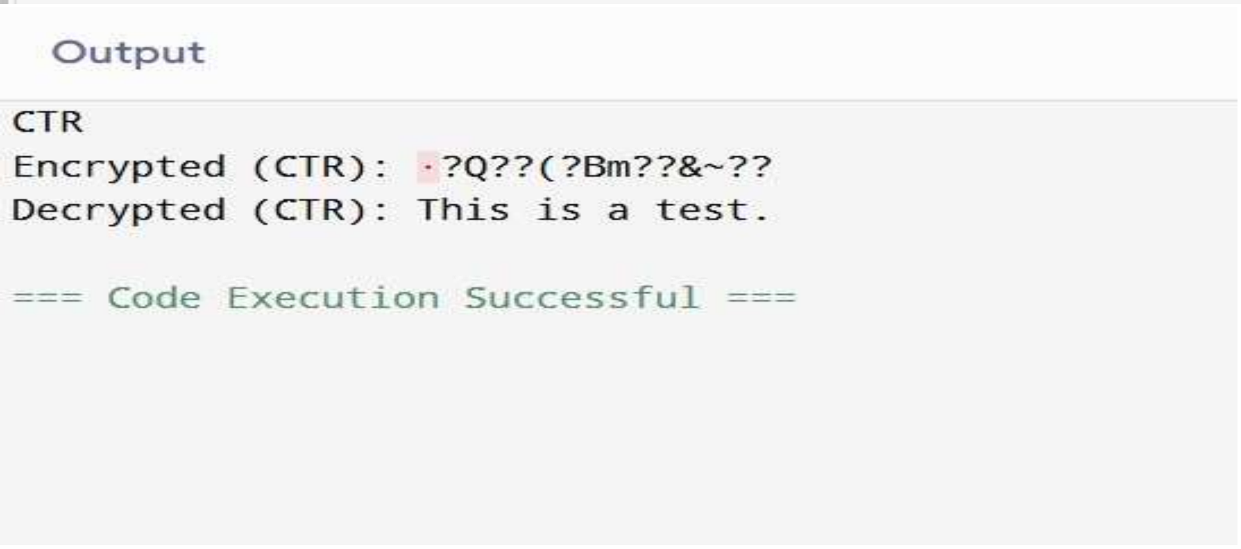
Bindu



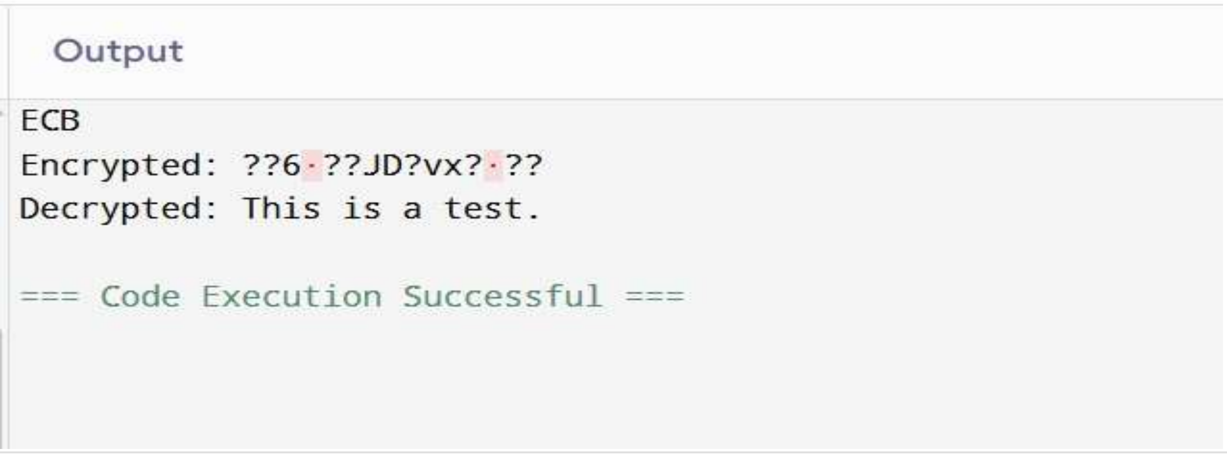
```
CBC
Encrypted (CBC): ·??h??4~?Hr?_?
Decrypted (CBC): This is a test.

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



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

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

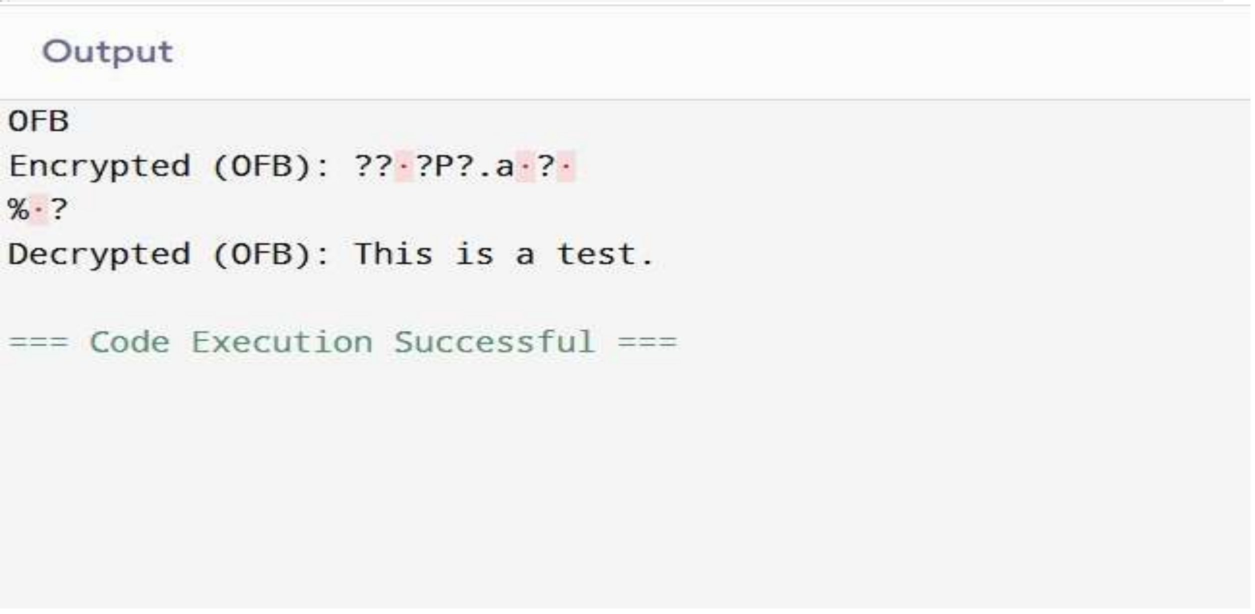


```
CTR
Encrypted (CTR): ·?Q??(?Bm??&~??
Decrypted (CTR): This is a test.

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



Output

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

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



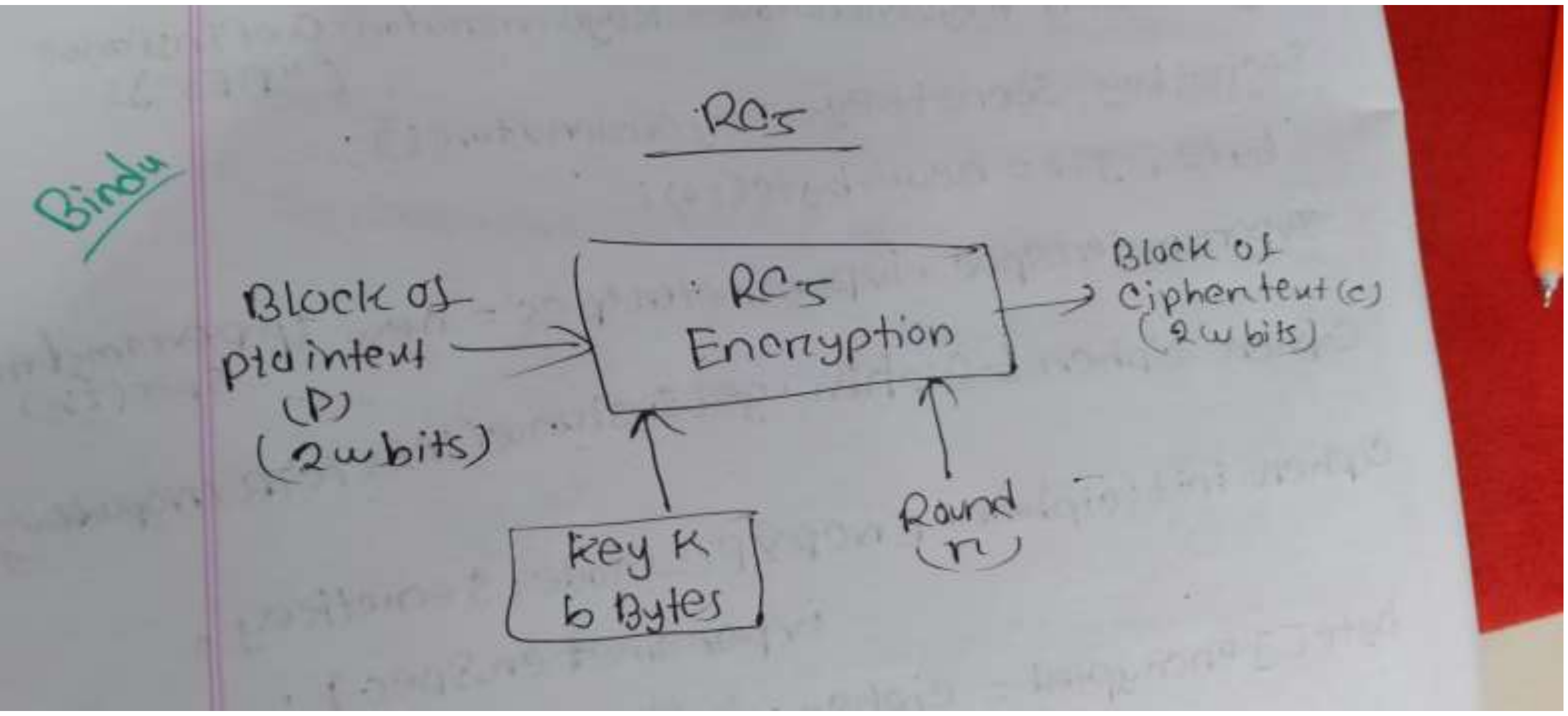
Output

```
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);
    }
}
```


Bindu
RC5
Block of plaintext (P) (2w bits)
RC5 Encryption
Block of ciphertext (c) (2w bits)
Key K b Bytes
Round (r)