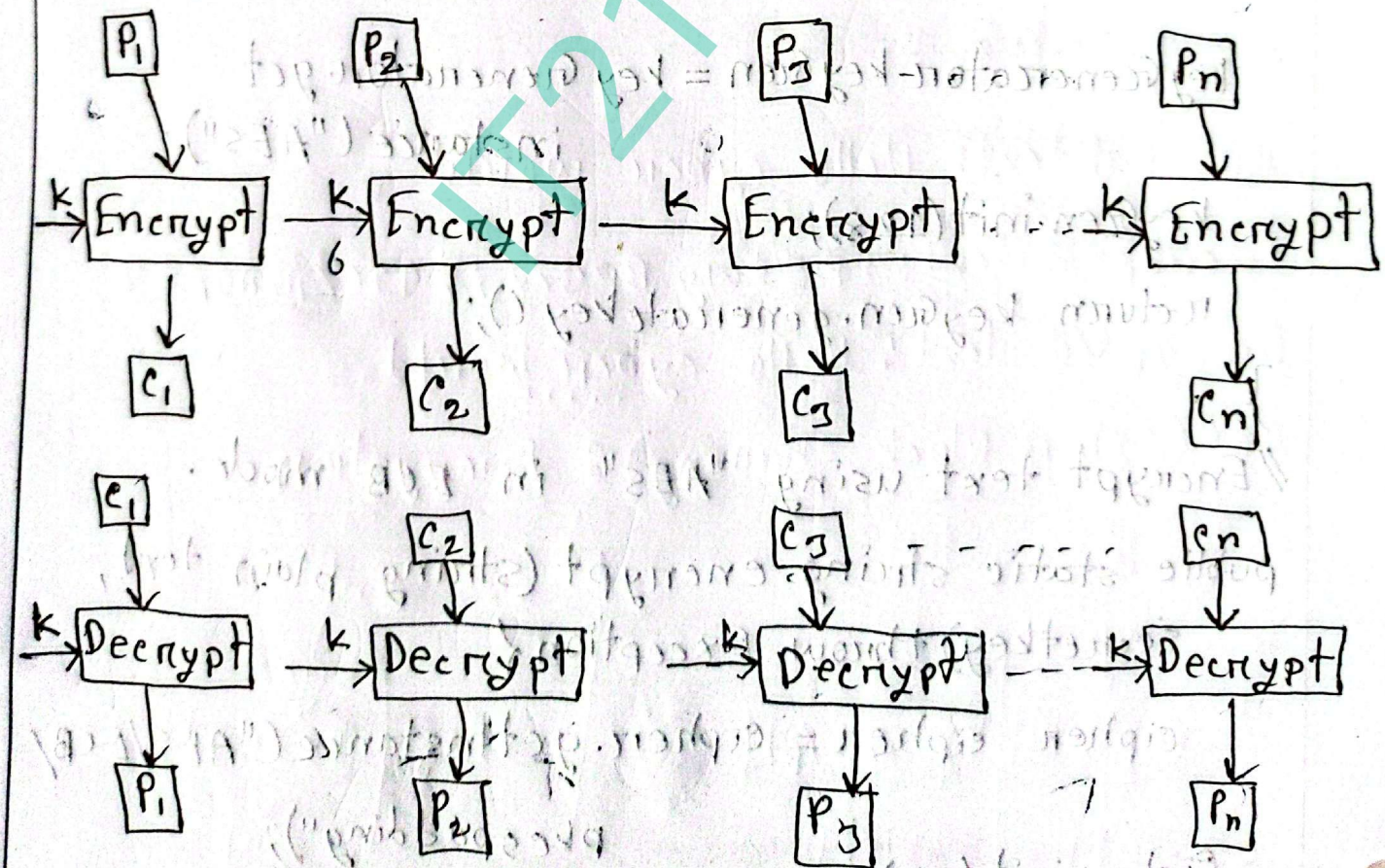
ID: IT-21024 (Md. Shahidul Islam)

*Assignment on: Modes of operation and RC5- Block diagram and java implementation and output- scanned pdf via Github.

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

Block diagram:



Java Implementation of ECB:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.util.Base64;
public class ECBmodeExample{
//Generate a sample AES key (128 bit)
  public static SecretKey generateAESkey () 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 plain text,
    SecretKey) throw, Exception{
    Cipher cipher = cipher.getInstance ("AES/ECB/
                                    PKCSPadding");
  cipher.init (cipher.ENCRYPT.mode, secret);
```

```
byte[] encryptBytes = cipher.doFinal(plaintext.get
                                    bytes());
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/PKCS5
                                    Padding");
 cipher.init(cipher.DECRYPTmode, secretkey);
 byteII decryptedBytes = cipher.doFinal(Base64.get
        Decoder).decode(encryptedText));
   return new string(decrypted Bytes);
//Example
Public static void main(string [] args){
 try {
     secretkey = generateABSkey();
     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);
}
catch (Exception e){
    e.printstackTrace();
}
}
}
```

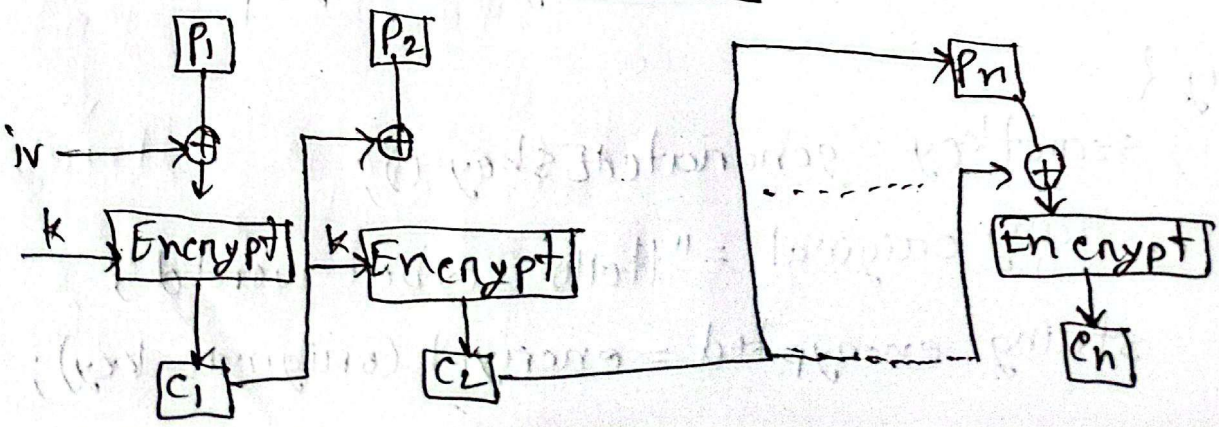

Output:

Original text: Hello cyber world

Encrypted text: fR0twy BP3ex1V41eZMnop24

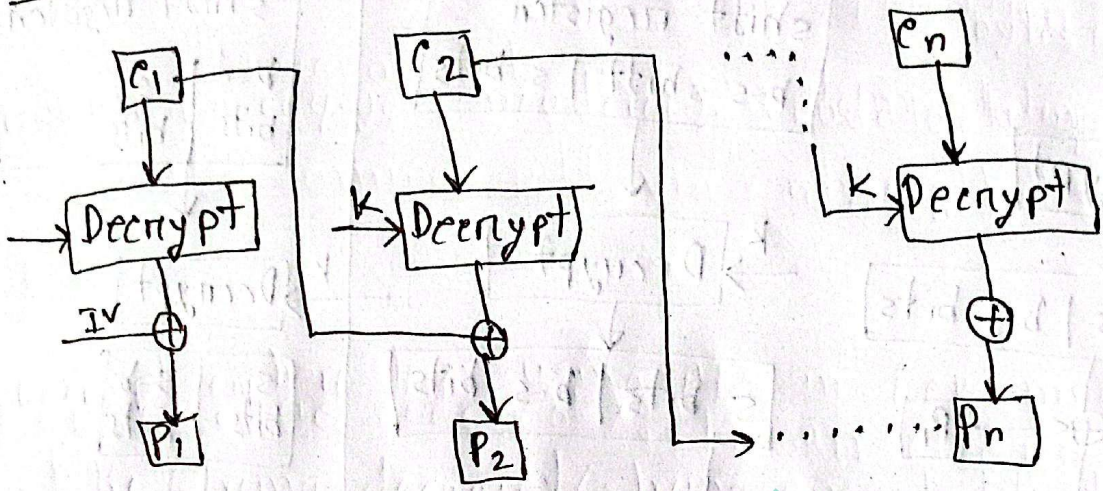
Decrypted text: Hello cyber world

CBC (cipher Block chaining): Encryption



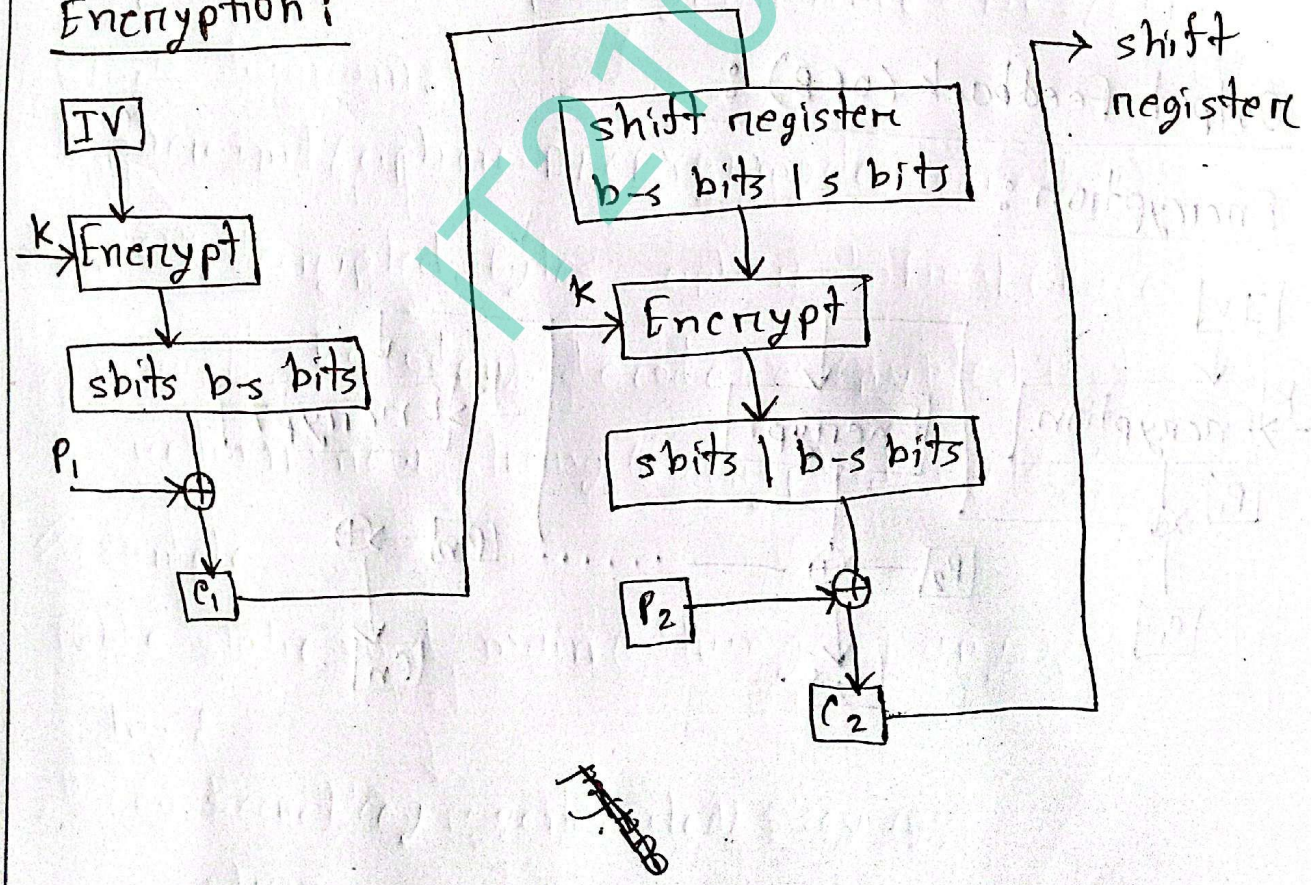
IT-21024

Decryption:



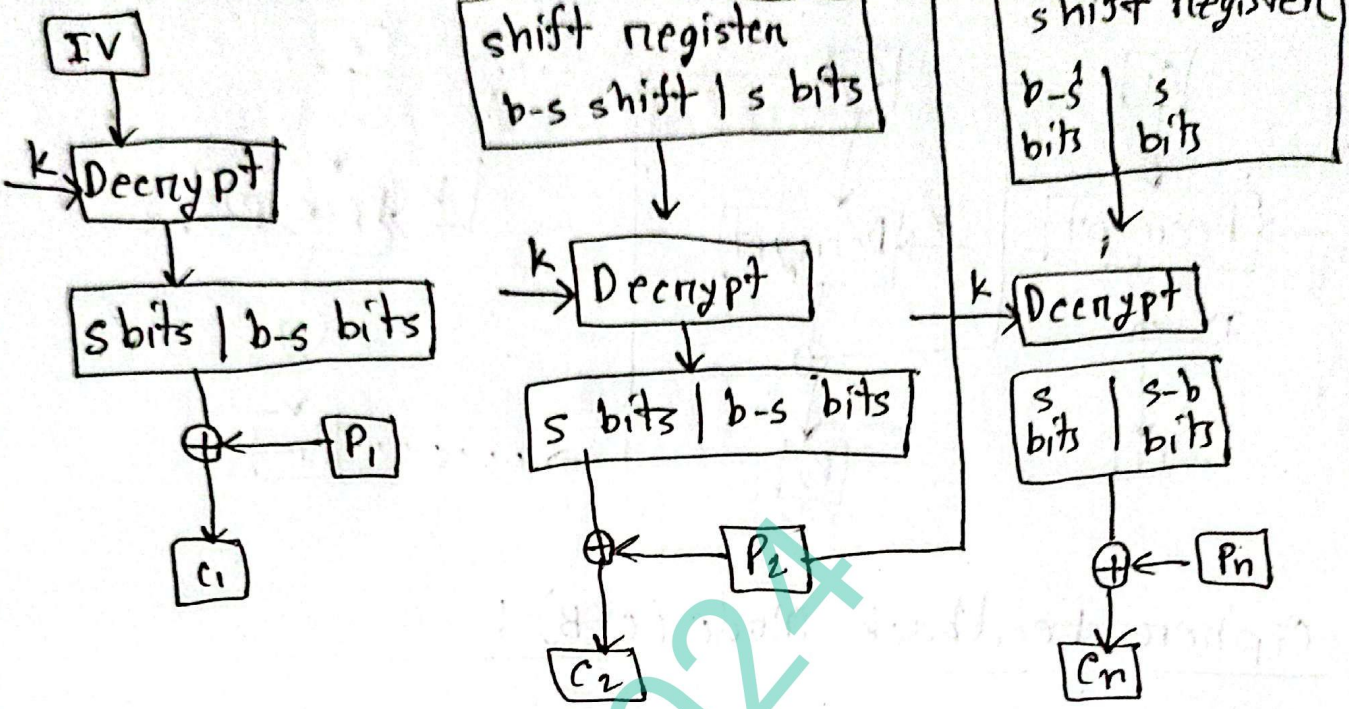
Cipher Feedback Mode (CFB):

Encryption:



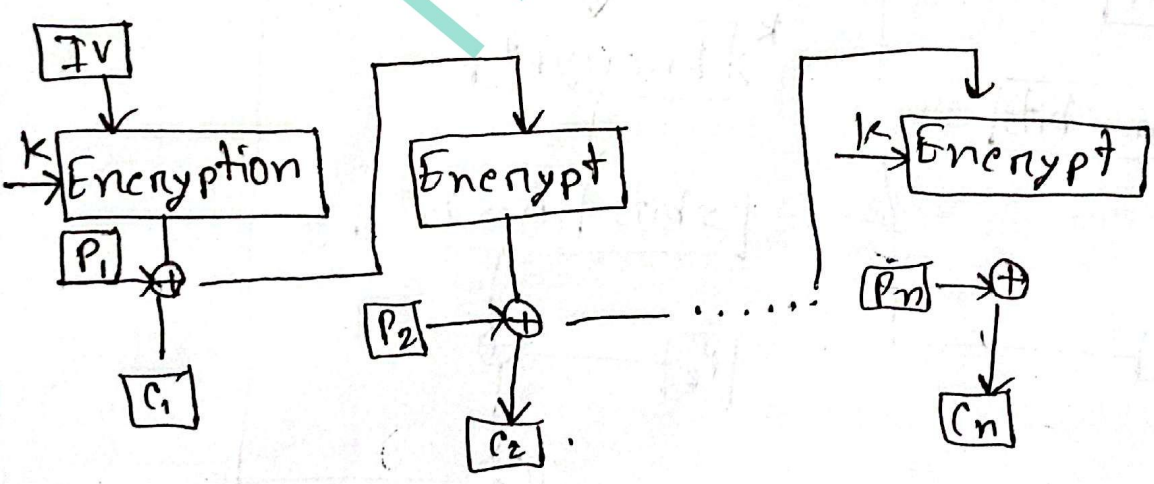
IT-21024

Decryption:

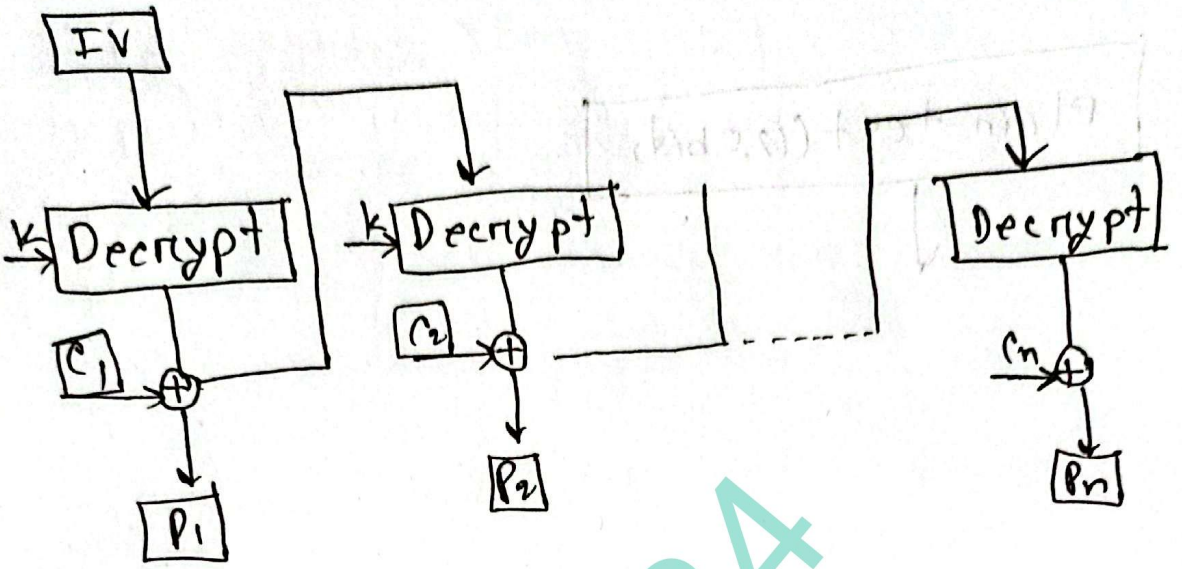


Output Feedback (OFB):

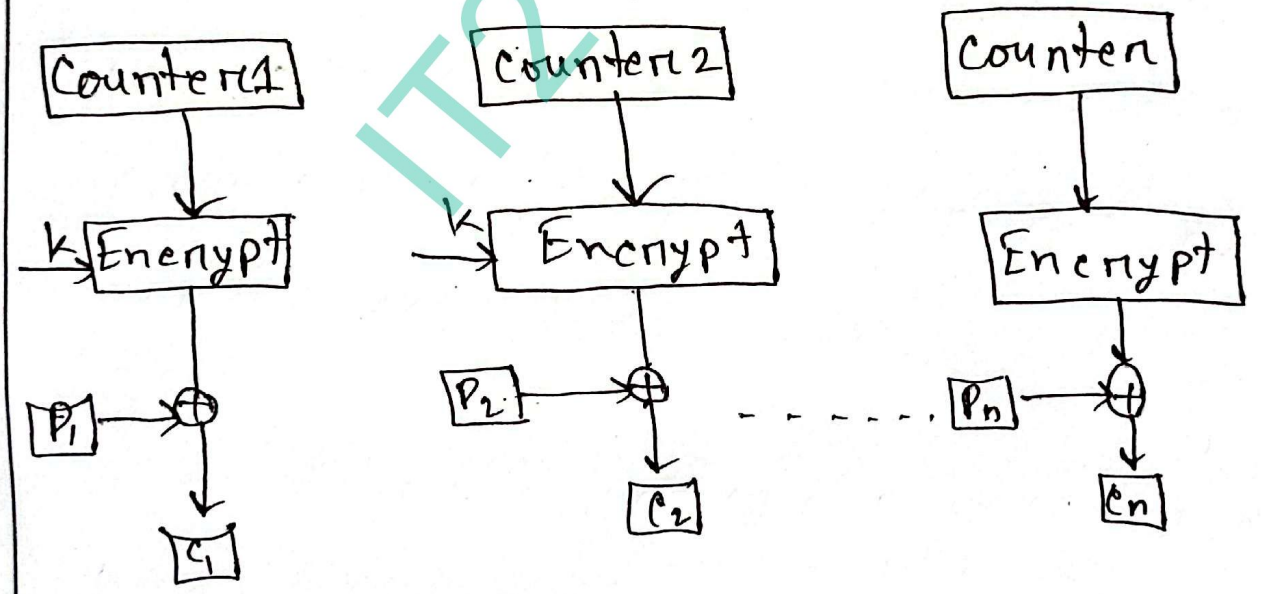
Encryption:



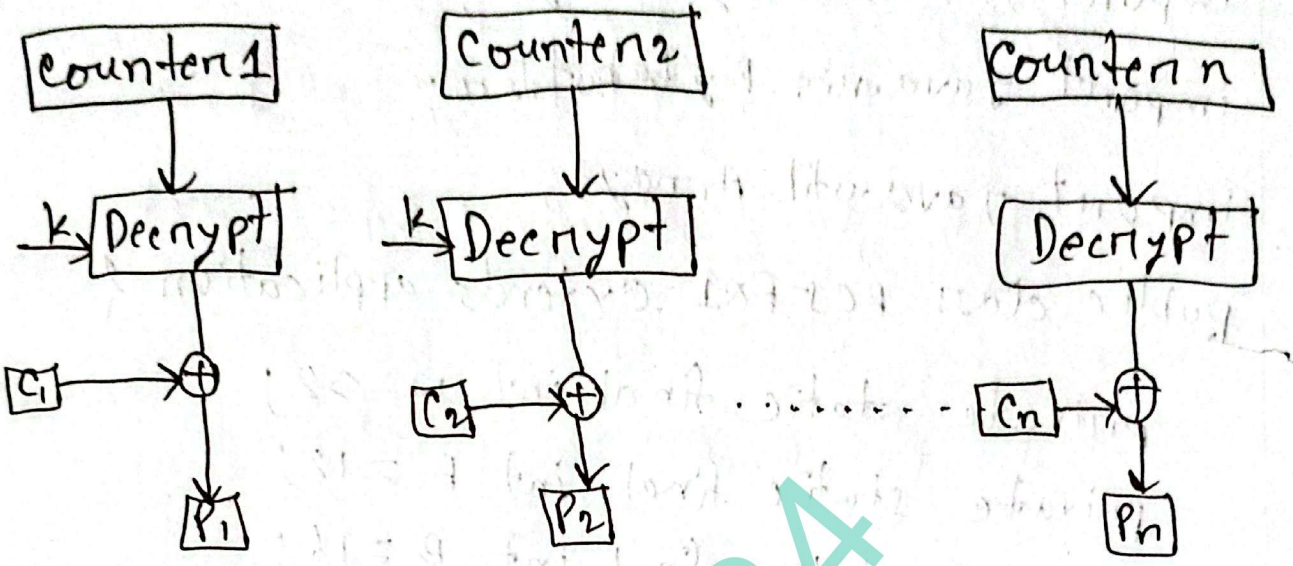
Decryption:



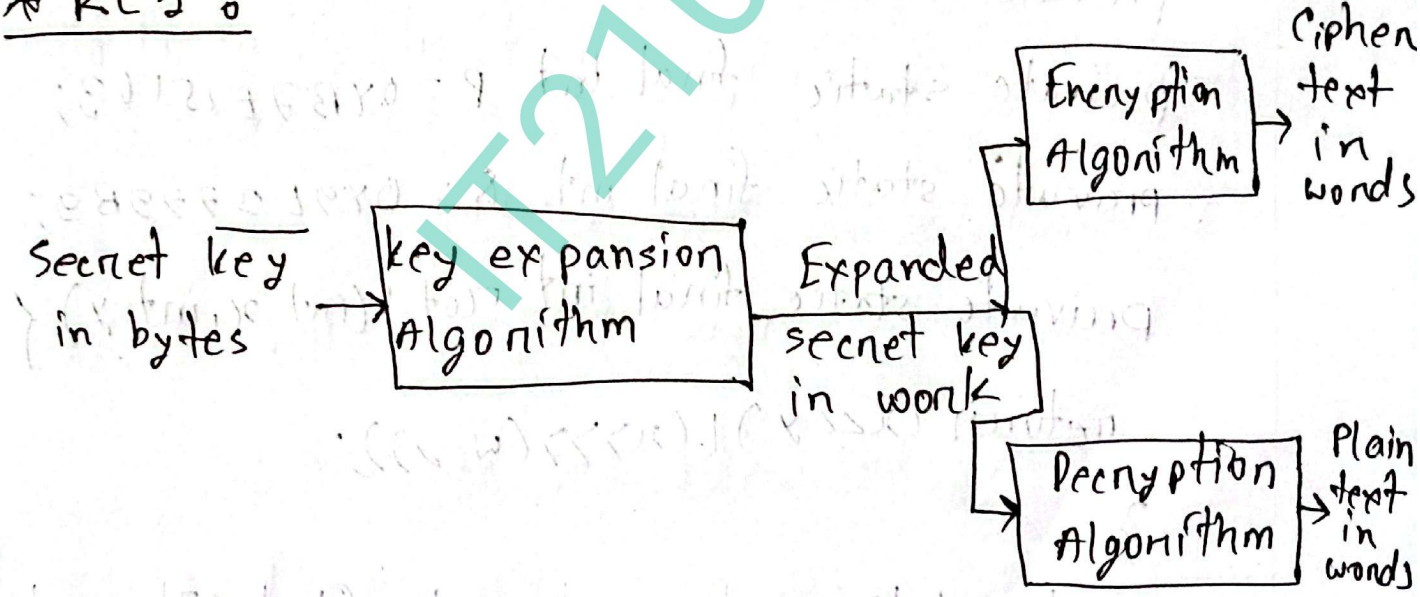
Counter mode (CTR): Encryption



Decryption:



* RC5 :



Code:

```
package org.example.rcbfx;
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.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 final 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<=*(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[i] = rotl(L[i]+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 = rot | (A^B, B) + s[2*i];
    B = rot | (B^A, A) + s[2*i+1];
}
data[0] = A;
data[1] = B;
}
Private static string encrypttext(string plaintext)
{   byte[] key = new byte[8];
    int[] s = new int[2*(R+1)];
    res key setup(key, s);
    byte[] plainbytes = plaintext.getBytes(standard
                        charaset: UTF = 8);
    int paddenlength = ((plainBytes.length+7)/8)*8;
    byte[] padded = Array.copy of(plainBytes,
                        paddellength);
```

```
StringBuffer ciphertext = new StringBuilder();
ByteBuffer.buffer = ByteBuffer.wrap(padded);
while (buffer.hasRemaining()) {
    int A = buffer.getInt();
    int B = buffer.getInt();
    int [] data = {A,B};
    desEncrypt(s,data);
    ciphertext.append(String.format("%.08x%
                .08x", data[0], data[1]));
}
    return ciphertext.toString().trim();
}
@Override
public void start(Stage stage){
    TextField input = new TextField();
    input.setPromptText("Enter English word to
                                encrypt");
    Button encryption = new Button("Encrypt");
```

```
TextArea output = new TextArea();
Output.setEditable(false);
output.setWrapText(true);
enencyptButton.setAction(e -> {
    string plaintext = input.getText();
    if (plaintext != null && !plaintext.isEmpty()) {
        string cipherntext = encyptText(plaintext);
    }
    else {
        output.setText("please enter some text");
    }
}
};
VBox.layout = new VBox(10 input enencyptButton,
                       output);
layout.setstyle("-fx-padding: 20;");
Scene scene = new Scene(layout, 500, 300);
stage.setTitle("RC5 Encryption 100");
stage.setScene(scene);
stage.show();
}
```

```
public static void main ("string [s] args) {
        launch (args)
    }
}
```

Output:

Input: hello world

Padding into 64 bit blocks

Block1: "hello wo" (8 bytes)

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

Encrypted output:

07c9fa2 b155 7b56 2493 583d 5f02 b8b3

Original: Hello world