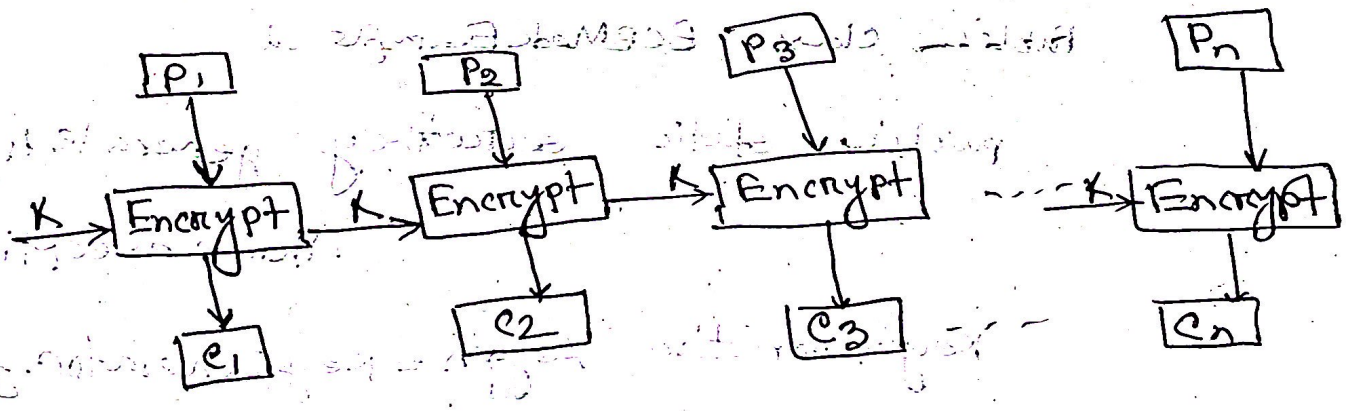
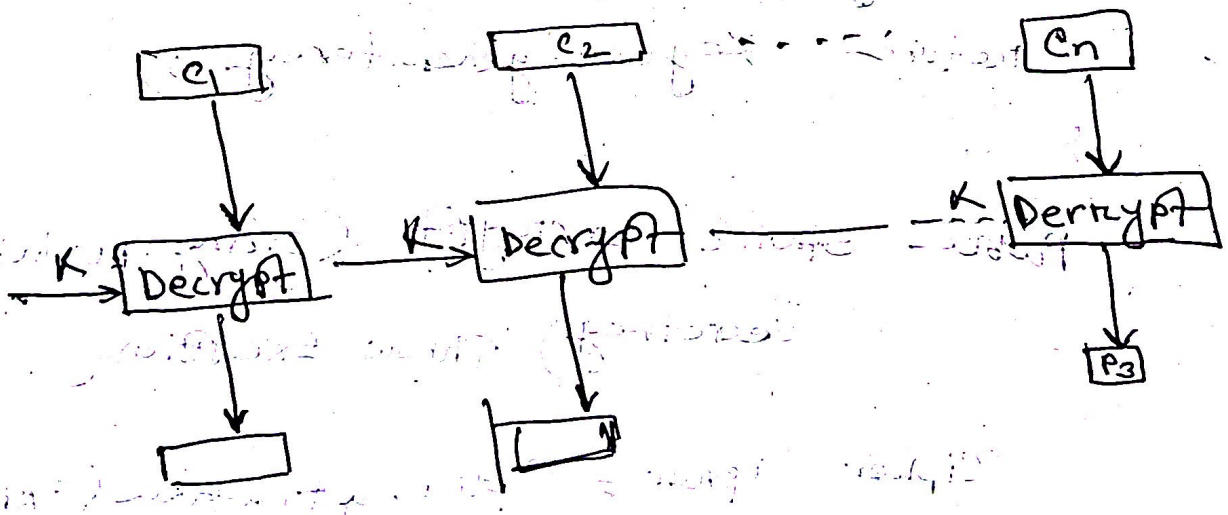
IT-21001

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

Block Diagram:





IT-21001

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 ECBModeExample2
    public static SecretKey generateAESkey()
                                Throw exception{
    KeyGeneration Keygen = KeyGenerator.get
                        instance("AES");
    KeyGen.init(128);
    return Keygen.generatekey();
}
Public static encrypt (String plaintext,
                SecretKey) Throw Exception{
    Cipher cipher = Cipher.getInstance("AES/ECB/
                        PKCS5Padding");
```

IT-21001

```
Cipher.init(cipher.ENCRYPT_Mode, secret);
byte[] encryptedBytes = cipher.doFinal(plaintext.getbytes);
return Base64.getEncoder().encodeString(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/PCKS5")
Cipher.init(cipher.DECRYPT_MODE, secretKey);
byte // decryptedBytes = cipher.doFinal(Base64.get
                                Decoder, decode(encryptedText));
return new string(decryptedBytes);
public static void main(String[] args) {
try{ secret key = generateAESKey();
```

IT-21001

```
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();
}
}
}
```

IT-21001

output:

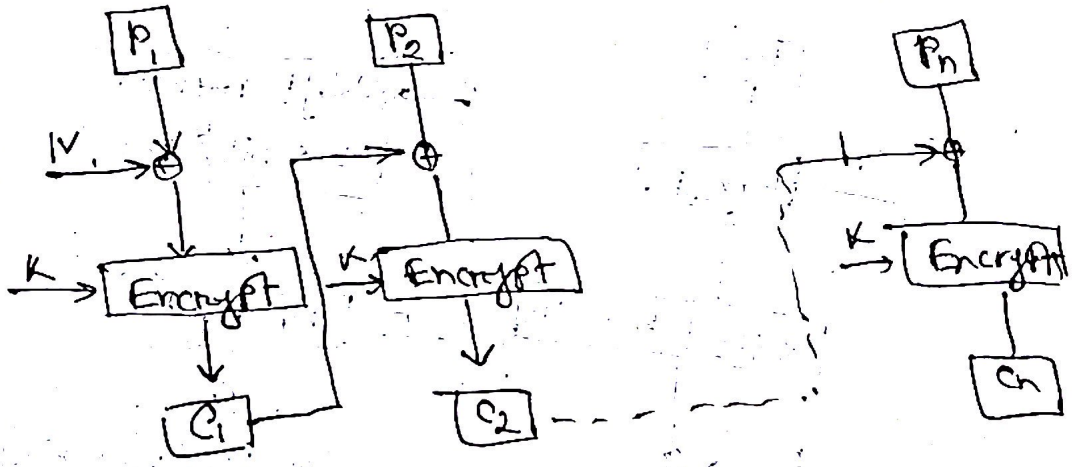
original Text : Hello cyber world !

Encrypted text: fRQt*yKBp 3AnX !V4lQWPlQRWRgU

Decrypted text: Hello cyber world !

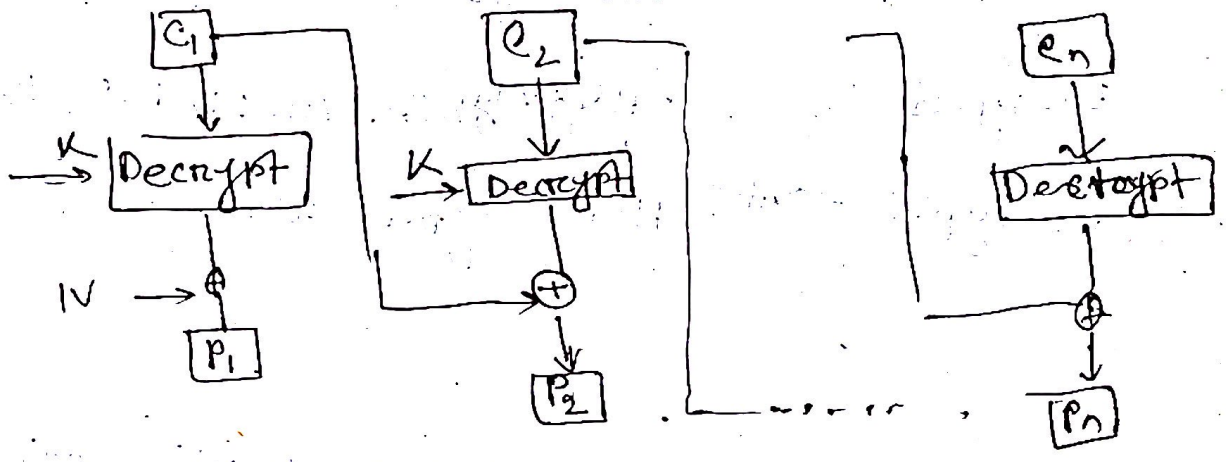
CBC (cipher Block chaining)

Encryption :



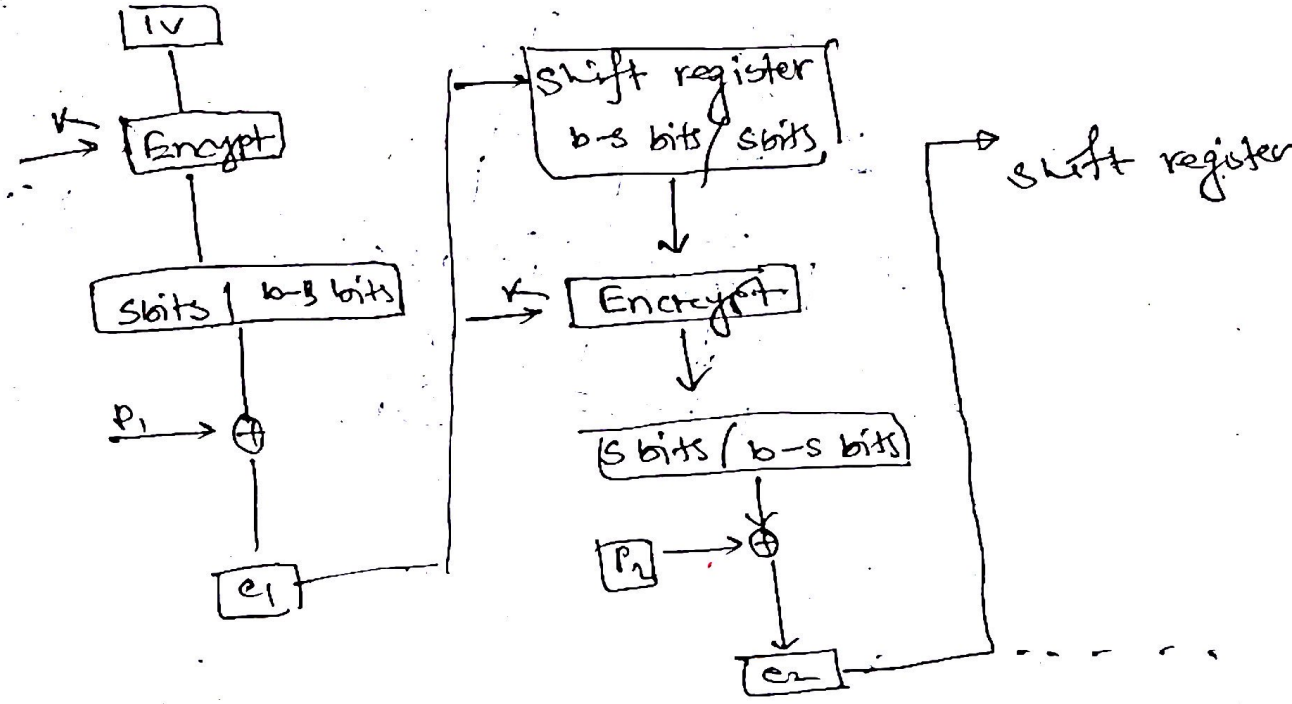
IT-21001

Decryption :



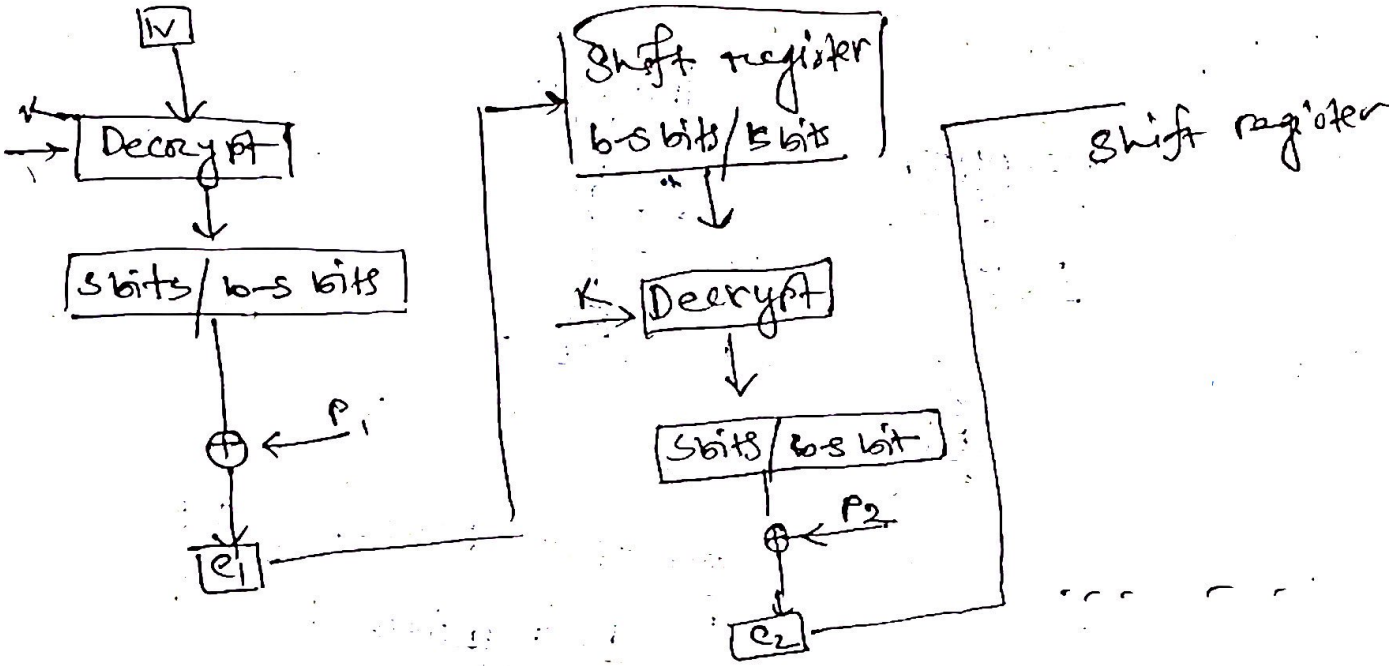
Cipher Feedback Mode (CFB)

Encryption



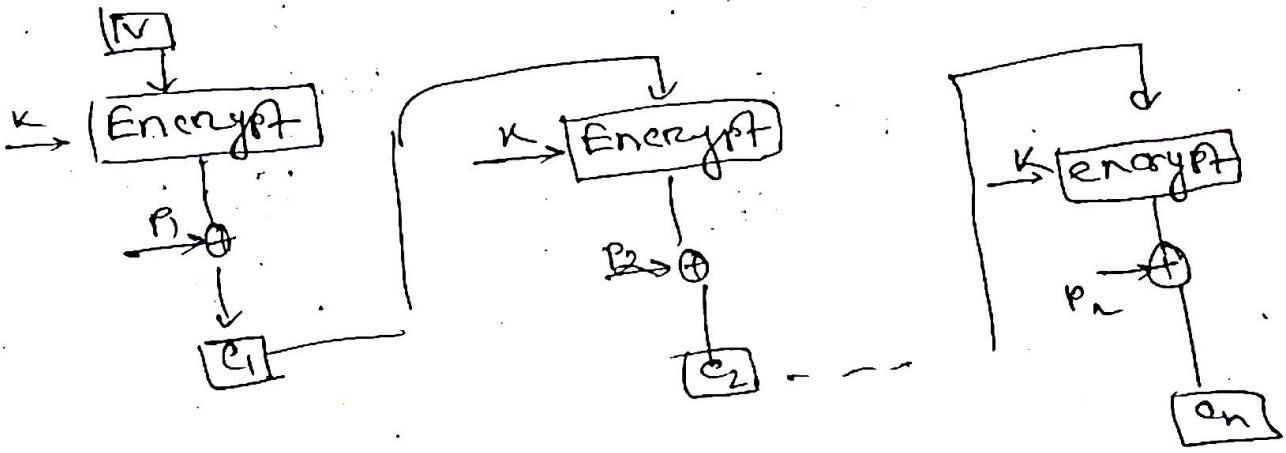
IT-21001

Decryption



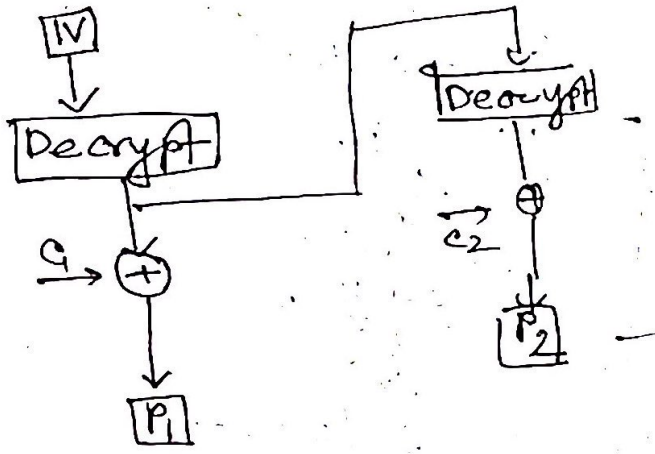
output feedback (OFB)

Encryption



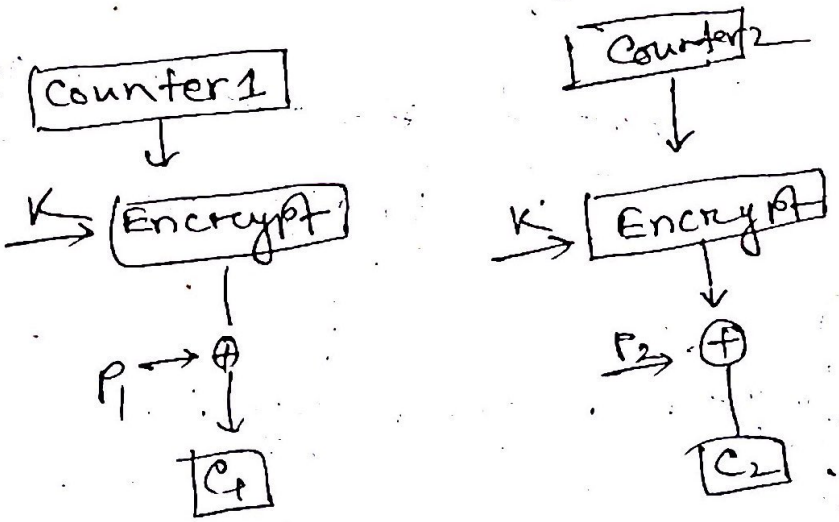
IT-21001

~~ECB~~ Decryption



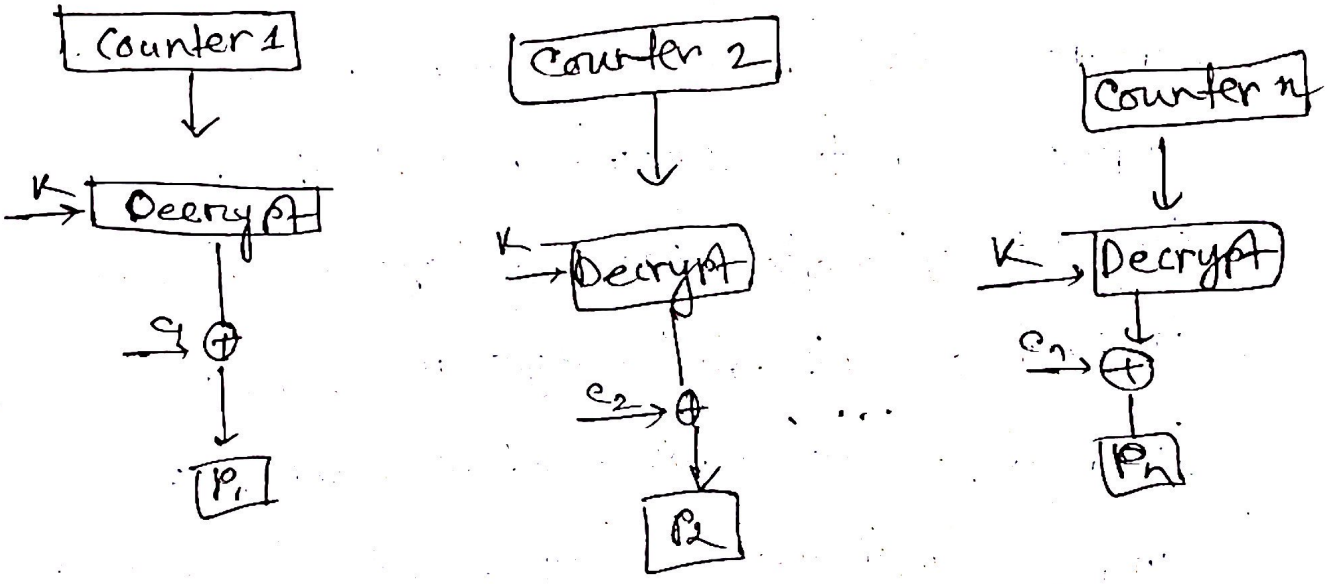
Counter Mode (CTR)

Encryption:

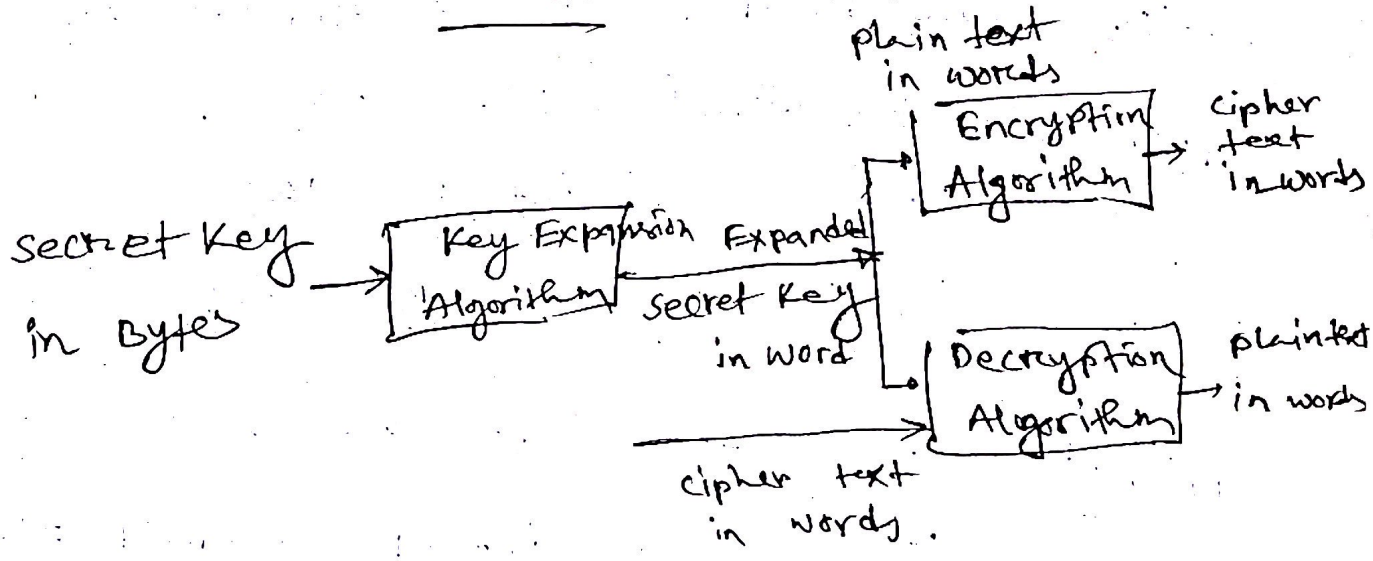


IT-21001

Decryption



RC5



IT-21001

Code:

```
package org.example.rsbfx;
import javafx.application.Application
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import java.nio.ByteBuffer;
import java.nio.stage.Stage;
import java.nio.channel.StandardChannel;
import java.util.Arrays;

public class RC5Fx1 extends Application
    private static final int w = 32;
            "       "   int R = 12;
            "       "   int B = 16;
    private "       "   int c = 4;
            "       "   int p = 0xB7E15163;
            "       "   int q = 0x9E3779B9;
            "       "   int rot(int x, int y) {
```

IT-21001

```
    return (x<<y) | (x>>> (w-y));
}
Private Static void rc5 key steup (byte [] key)
{
  private static void RC5 key setup (byte [] key)
    int [ ] = 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] + Q;
}
int A = 0, B = 0;
int i = 0, j = 0
Private static void rc5 Encrypt (int [] s, int [] data
{
    int A = data [0]
    int B = data [i]
```

IT-21001

```
A = A + S[0];
B = B + S[1];
for (int i=1; i<=R; ++i){
    A = not | (A^B; B) + S[2*i];
    B = not | (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)];
    rc5 key setup (key, 5);
byte[] plainBytes = plaintext.getBytes (standard
                    charsetn UTF = 8);
```

IT-21001

```
while (buffer.hasRemaining()){
    int A = buffer.getInt();
    int B = buffer.getInt();
    int[] data = {A, B};
rc5.Encrypt(s, data);
    ciphertext.append(String.format("%08x
                %08x", data[0], data[1]));
}
return ciphertext.toString().trim();
}
```

IT-21001

```
@ override
  public void start (Stage . stage){
Textfield input = new Textfield ();
input . setpromptText ("Enter English to Encrypt");
Button encrypt Button = new Button ("Encrypt");
 TextArea output = new . TextArea ();
 output . set Edittable (false);
  output . set wrapText (true);
encrypt Button . set on Action (e →{
 String plaintext = input . getText ();
if (plaintext ! = null && ! plainText . is Empty ()){
  String cipherText = encrypt . Text (plainText);
}
else
   { output . setText ("Please enter some text"))
   }
}
}
```

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

IT-21001

```
@Override
public void start (Stage stage){
TextField input = new TextField ();
input.setPromptText ("Enter English to encry
Button encryptButton = new Button ("Encrypt");
TextArea output = new TextArea ();
output.setWrapText (true);
encryptButton.setOnAction (e -> {
String plainText = input.getText ();
if (plainText != null && !plainText.isEmpty ()) {
String cipherText = encrypt.Text (plainTe
}
else
{
output.setText ("Please enter some text")
}
}
```

IT-21001

```
VBox layout = new VBox (10, input encryptButton,
                        output);
layout.setStyle ("-fx-padding: 20;");
Scene scene = new Scene (layout, 500, 300);
Stage setScence (scene);
Stage.show();
Public static void main (String [] args) {
        launch (args)
    }
}
```

output

input : hello world

padding into 64 bit blocks

Block 1 : "hellow " (8 bytes)

Block 2 : " rld 10101010 " (Padded)

Encrypted output