**Github link:**[**https://github.com/TataChetanasree/Cryptography**](https://github.com/TataChetanasree/Cryptography)

**AES Code**

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

import javax.crypto.BadPaddingException;

import java.security.SecureRandom;

import java.util.Base64;

import java.util.Scanner;

public class SymmetricCipherAESInput {

public static SecretKey getKeyFromUser(String keyStr) {

byte[] keyBytes = keyStr.getBytes();

byte[] keyBytesPadded = new byte[16]; // AES-128 requires 16 bytes

System.arraycopy(keyBytes, 0, keyBytesPadded, 0, Math.min(keyBytes.length, 16));

return new SecretKeySpec(keyBytesPadded, "AES");

}

public static IvParameterSpec generateIV() {

byte[] iv = new byte[16];

new SecureRandom().nextBytes(iv);

return new IvParameterSpec(iv);

}

public static String encrypt(String plaintext, SecretKey key, IvParameterSpec iv) throws Exception {

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

cipher.init(Cipher.ENCRYPT\_MODE, key, iv);

byte[] encrypted = cipher.doFinal(plaintext.getBytes());

byte[] ivAndEncrypted = new byte[iv.getIV().length + encrypted.length];

System.arraycopy(iv.getIV(), 0, ivAndEncrypted, 0, 16);

System.arraycopy(encrypted, 0, ivAndEncrypted, 16, encrypted.length);

return Base64.getEncoder().encodeToString(ivAndEncrypted);

}

public static String decrypt(String encryptedBase64, SecretKey key) throws Exception {

byte[] ivAndEncrypted = Base64.getDecoder().decode(encryptedBase64);

byte[] iv = new byte[16];

byte[] encrypted = new byte[ivAndEncrypted.length - 16];

System.arraycopy(ivAndEncrypted, 0, iv, 0, 16);

System.arraycopy(ivAndEncrypted, 16, encrypted, 0, encrypted.length);

IvParameterSpec ivSpec = new IvParameterSpec(iv);

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

cipher.init(Cipher.DECRYPT\_MODE, key, ivSpec);

byte[] decrypted = cipher.doFinal(encrypted);

return new String(decrypted);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.print("Enter message to encrypt (AES): ");

String message = scanner.nextLine();

System.out.print("Enter AES key (any string, 16 chars recommended): ");

String keyInput = scanner.nextLine();

SecretKey key = getKeyFromUser(keyInput);

IvParameterSpec iv = generateIV();

String encrypted = encrypt(message, key, iv);

System.out.println("Encrypted (AES): " + encrypted);

String decrypted = decrypt(encrypted, key);

System.out.println("Decrypted (AES): " + decrypted);

} catch (BadPaddingException e) {

System.out.println("Decryption failed: possibly wrong key or corrupted data.");

} catch (Exception e) {

e.printStackTrace();

}

scanner.close();

}

}

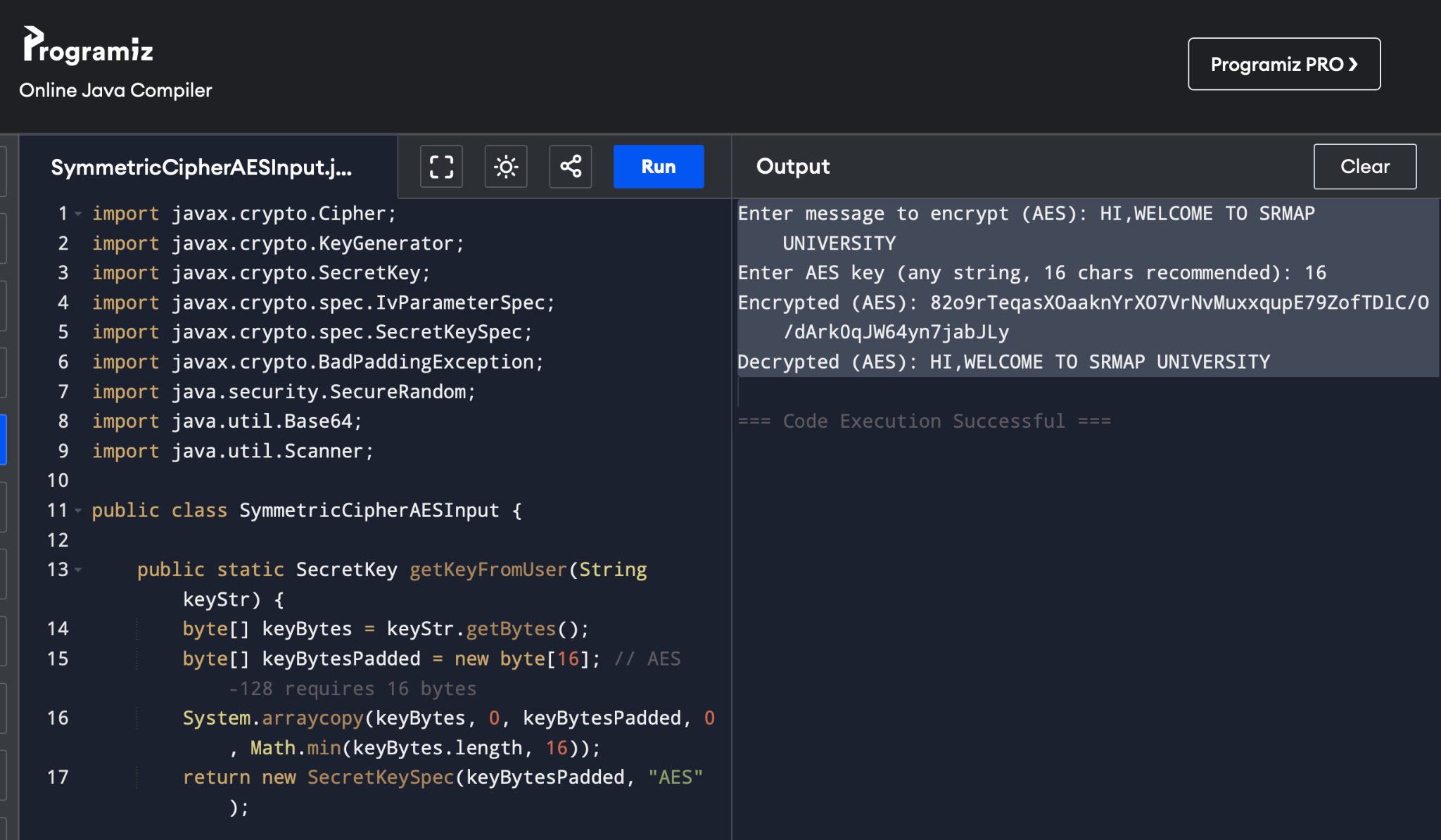
OUTPUT:

Enter message to encrypt (AES): HI,WELCOME TO SRMAP UNIVERSITY

Enter AES key (any string, 16 chars recommended): 16

Encrypted (AES): 82o9rTeqasXOaaknYrXO7VrNvMuxxqupE79ZofTDlC/O/dArk0qJW64yn7jabJLy

Decrypted (AES): HI,WELCOME TO SRMAP UNIVERSITY



**RC4 Code**

AP25122050006

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

public class SymmetricCipherRC4 {

public static String encrypt(String plaintext, byte[] key) throws Exception {

SecretKey secretKey = new SecretKeySpec(key, "RC4");

Cipher cipher = Cipher.getInstance("RC4");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

byte[] encrypted = cipher.doFinal(plaintext.getBytes());

return Base64.getEncoder().encodeToString(encrypted);

}

public static String decrypt(String ciphertextBase64, byte[] key) throws Exception {

SecretKey secretKey = new SecretKeySpec(key, "RC4");

Cipher cipher = Cipher.getInstance("RC4");

cipher.init(Cipher.DECRYPT\_MODE, secretKey);

byte[] ciphertext = Base64.getDecoder().decode(ciphertextBase64);

byte[] decrypted = cipher.doFinal(ciphertext);

return new String(decrypted);

}

public static void main(String[] args) throws Exception {

String message = "Hello, this is a secret message!";

byte[] key = "secretkey1234567".getBytes(); // RC4 key (length can vary, but 16 bytes is common)

String encrypted = encrypt(message, key);

System.out.println("Encrypted (RC4): " + encrypted);

String decrypted = decrypt(encrypted, key);

System.out.println("Decrypted (RC4): " + decrypted);

}

}

OUTPUT:

Encrypted (RC4): XTnbI2ljUQTU1y5sc39PFYWGdLkni8PAeXefU4cB/qM=

Decrypted (RC4): Hello, this is a secret message!

