Department Of Computer Science

### Gujarat University



**Certificate**

**R**oll No: 23 Seat No: \_\_\_\_\_\_

This is to certify that Ms. \_\_\_\_DEVANGI BHARATBHAI PABARI\_\_\_ student of MCA Semester – V, has duly completed her term work for the semester ending in December 2021, in the subject of \_\_NETWORK SECURITY\_\_\_ towards partial fulfillment of her Degree of Masters in Computer Science & Application.

14/12/2021

Date of Submission Internal Faculty

Head of Department

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NO. | TITLE | PAGE NO. | DATE | SIGN |
| **1** | **Implement Caesar Cipher** | **1** | **14/12/2021** |  |
| **2** | **Implement Substitution Cipher(mono-alphabatic)** | **2** | **14/12/2021** |  |
| **3** | **Implement Transposition Cipher** | **4** | **14/12/2021** |  |
| **4** | **Implement ONE TIME PAD (OTP)** | **9** | **14/12/2021** |  |
| **5** | **Implement P-Box(character based)** | **12** | **14/12/2021** |  |
| **6** | **Implement S-Box(character based)** | **14** | **14/12/2021** |  |
| **7** | **Implement DES/3DES/AES using java APIs**  **(use different modes+ Base 64 encoding)** | **16** | **14/12/2021** |  |
|  | 1. **Generate Keys** |  |  |  |
|  | 1. **generate Initialization Vector(IV)** |  |  |  |
|  | 1. **Select Mode(ECB/CBC/CFB)** |  |  |  |
|  | 1. **Cipher Text -> Base64 Encoding** |  |  |  |
| **8** | **Message Digest (MD5, SHA-1 hash, SHA-256 hash)** | **20** | **14/12/2021** |  |
| **9** | **RSA** | **24** | **14/12/2021** |  |
| **10** | **RSA using API** | **27** | **14/12/2021** |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

----------------------------------------------------------------------------------------------------------------

QUESTION 1 :

Implement Caesar Cipher

----------------------------------------------------------------------------------------------------------------

import java.util.Scanner;

public class CeaserCipher {

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

public static String doEncription(int key , String plaintext){

plaintext =plaintext.toLowerCase();

String Ciphertext="";

for(int i=0;i<plaintext.length();i++){

int val = ALPHABET.indexOf(plaintext.charAt(i));

int res = (val+key)%26;

Ciphertext += ALPHABET.charAt(res);

}

return Ciphertext;

}

public static String doDecription(int key, String ciphertext){

String plaintext="";

for(int i=0;i<ciphertext.length();i++){

int val = ALPHABET.indexOf(ciphertext.charAt(i));

int res = (val - key) % 26;

if (res < 0)

{

res = ALPHABET.length() + res;

}

plaintext += ALPHABET.charAt(res);

}

return plaintext;

}

public static void main(String[] args) {

Scanner sc =new Scanner(System.in);

String plaintext =sc.nextLine();

String encryptedText= doEncription(3, plaintext);

String decryptedText= doDecription(3, encryptedText);

System.out.println("Encriptedtext :"+encryptedText);

System.out.println("Decrypted text :"+decryptedText);

}

}

---------

Output

----------

Enter Plaintext :hello

Encrypted text :khoor

Decrypted text :hello

----------------------------------------------------------------------------------------------------------------

QUESTION 2 :

Implement Substitution Cipher (mono-alphabetic)

----------------------------------------------------------------------------------------------------------------

import java.util.Scanner;

public class SubstitutionCipher {

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

public static final String REPLACINGALPHABET = "uivejfamwgxnycozksdtprbqhl";

public static String doEncription(String plaintext){

plaintext =plaintext.toLowerCase();

String Ciphertext="";

for(int i=0;i<plaintext.length();i++){

int val = ALPHABET.indexOf(plaintext.charAt(i));

Ciphertext += REPLACINGALPHABET.charAt(val);

}

return Ciphertext;

}

public static String doDecription( String ciphertext){

String plaintext="";

for(int i=0;i<ciphertext.length();i++){

int val = ALPHABET.indexOf(ciphertext.charAt(i));

plaintext += REPLACINGALPHABET.charAt(val);

}

return plaintext;

}

public static void main(String[] args) {

Scanner sc =new Scanner(System.in);

String plaintext =sc.nextLine();

String encryptedText= doEncription(plaintext);

String decryptedText= doDecription(encryptedText);

System.out.println("Encriptedtext :"+encryptedText);

System.out.println("Decrypted text :"+decryptedText);

}

}

----------

Output

----------

Enter Plaintext : hello

Encriptedtext :mjnno

----------------------------------------------------------------------------------------------------------------

QUESTION 3 :

Implement Transposition Cipher

----------------------------------------------------------------------------------------------------------------

import java.util.Arrays;

import java.util.Scanner;

public class TranspositionCipher {

public static final String ALPHA = "abcdefghijklmnopqrstuvwxyz";

public static char ALPHABET[] = ALPHA.toCharArray();

public static String doEncryption(char[] plaintext, int[] key) {

String encrypttext = "";

int row = findLength(plaintext, key);

int col = key.length;

char[][] encrArray = new char[row][col];

// first fill whole array with '\*'

for(int x=0;x<encrArray.length;x++)

for(int y=0;y<encrArray[x].length;y++)

encrArray[x][y] = '\*';

// fill array with plain text

int k = 0;

for (int i = 0; i< row; i++) {

for (int j = 0; j < col; j++) {

if(plaintext.length>k){

encrArray[i][j] = plaintext[k];

k++;

}

}

}

// read array key wise

for (int i = 0; i<key.length; i++) {

int l = key[i];

for (int j = 0; j < row; j++) {

encrypttext += encrArray[j][l];

}

}

return encrypttext;

}

public static String doDecryption(char[] encriptedText, int[] keyArray) {

int row = findLength(encriptedText, keyArray);

int col = keyArray.length;

char[][] decriptionArray = new char[row][col]; // 3\*4 array

String resultingDecryptionText = new String();

int o=0;

for(int k=0; k<keyArray.length;k++)

{

int i=keyArray[k];

for (int j = 0; j < row; j++) {

decriptionArray[j][i] = encriptedText[o];

o++;

}

}

for (int i = 0; i< row; i++) {

for (int j = 0; j < col; j++) {

resultingDecryptionText += decriptionArray[i][j];

}

}

return resultingDecryptionText;

}

public static int findLength(char[] plaintext, int[] key) {

// System.out.println(17%4); plaintext/key if we get precision value than it

// include 1 row more

int ptLength = plaintext.length; // 17

int klength = key.length; // 4

int col = klength;

int row = 0;

if (ptLength>klength) {

row = ptLength / klength;

if (ptLength % klength != 0) {

row += 1;

}

}

return row;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

char[] plaintext, key;

System.out.println("Enter Plaintext");

plaintext = sc.nextLine().toCharArray(); // hear replacing space with '\*'

System.out.println("Enter Key");

key = sc.nextLine().toCharArray();

int KeyToIntInAsending[] = new int[key.length];

int k = 0;

for (int i = 0; i<ALPHABET.length; i++) {

for (int j = 0; j <key.length; j++) {

if (ALPHABET[i] == key[j]) {

KeyToIntInAsending[k] = j;

k++;

break;

}

}

}

String encriptedString = doEncryption(plaintext, KeyToIntInAsending);

System.out.println("EncriptedString :" + encriptedString);

String decriptedString = doDecryption(encriptedString.toCharArray(), KeyToIntInAsending);

System.out.println("DecriptedString :" + decriptedString);

}

}

--------

Ouput

--------

Enter Plaintext

sanfoundry

Enter Key

hack

EncriptedString :auynn\*sorfd\*

DecriptedString :sanfoundry\*\*

----------------------------------------------------------------------------------------------------------------

QUESTION 4 :

Implement ONE TIME PAD (OTP)

----------------------------------------------------------------------------------------------------------------

import java.io.\*;

import java.util.Scanner;

public class OneTimePad {

public static String stringEncryption(String text,String key)

{

String cipherText = "";

int cipher[] = new int[key.length()];

for (int i = 0; i<key.length(); i++)

{

cipher[i] = text.charAt(i) - 'A' + key.charAt(i)- 'A';

}

for (int i = 0; i<key.length(); i++)

{

if (cipher[i] > 25)

{

cipher[i] = cipher[i] - 26;

}

}

for (int i = 0; i<key.length(); i++)

{

int x = cipher[i] + 'A';

cipherText += (char)x;

}

return cipherText;

}

public static String stringDecryption(String s,String key)

{

String plainText = "";

int plain[] = new int[key.length()];

for (int i = 0; i<key.length(); i++)

{

plain[i]= s.charAt(i) - 'A' - (key.charAt(i) - 'A');

}

for (int i = 0; i<key.length(); i++)

{

if (plain[i] < 0)

{

plain[i] = plain[i] + 26;

}

}

for (int i = 0; i<key.length(); i++)

{

int x = plain[i] + 'A';

plainText += (char)x;

}

return plainText;

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter Plaintext :");

String plainText = sc.nextLine();

System.out.print("Enter key :");

String key = sc.nextLine();

String encryptedText =stringEncryption(plainText.toUpperCase(), key.toUpperCase());

System.out.println("Cipher Text - "+ encryptedText);

System.out.println("Message - " + stringDecryption(encryptedText,key.toUpperCase()));

}

}

---------

output

----------

Enter Plaintext :hello

Enter key :world

Cipher Text - DSCWR

Message – HELLO

----------------------------------------------------------------------------------------------------------------

QUESTION 5 :

Implement P-Box (character based) (Eg. cat → atc)

----------------------------------------------------------------------------------------------------------------

import java.util.\*;

public class Pbox extends Thread {

public static ArrayList list = new ArrayList();

public static void main(String args[]) {

Pboxpbox = new Pbox();

pbox.start();

Pbox s2 = new Pbox();

s2.start();

}

public void run(){

Random rand = new Random();

System.out.println("Please enter the string ");

Scanner input = new Scanner(System.in);

String original=input.nextLine();

permute(original);

int index = rand.nextInt(list.size());

System.out.println("SBox of " + original + " is :" + list.get(index));

}

public static void permute( String input)

{

int inputLength = input.length();

boolean[ ] used = new boolean[ inputLength ];

StringBufferoutputString = new StringBuffer();

char[ ] in = input.toCharArray( );

doPermute( in, outputString, used, inputLength, 0 );

}

public static void doPermute( char[ ] in, StringBufferoutputString,

boolean[ ] used, int inputLength, int level)

{

if( level == inputLength) {

list.add(outputString.toString());

// System.out.println( outputString.toString());

return;

}

for( inti = 0; i<inputLength; ++i )

{

if( used[i] ) continue;

outputString.append( in[i] );

used[i] = true;

doPermute( in, outputString, used, inputLength, level + 1 );

used[i] = false;

outputString.setLength( outputString.length() - 1 );

}

}

}

---------

Output

----------

Please enter the string

Please enter the string

hello

world

SBox of hello is :drlow

SBox of world is :olrwd

----------------------------------------------------------------------------------------------------------------

QUESTION 6 :

Implement S-Box (character based) (Eg. cat → xma) (substitution )

----------------------------------------------------------------------------------------------------------------

import java.util.Scanner;

public class SubstitutionCipher {

public static final String ALPHABET = "abcdefghijklmnopqrstuvwxyz";

public static final String REPLACINGALPHABET = "uivejfamwgxnycozksdtprbqhl";

public static String doEncription(String plaintext){

plaintext =plaintext.toLowerCase();

String Ciphertext="";

for(int i=0;i<plaintext.length();i++){

int val = ALPHABET.indexOf(plaintext.charAt(i));

Ciphertext += REPLACINGALPHABET.charAt(val);

}

return Ciphertext;

}

public static String doDecription( String ciphertext){

String plaintext="";

for(int i=0;i<ciphertext.length();i++){

int val = ALPHABET.indexOf(ciphertext.charAt(i));

plaintext += REPLACINGALPHABET.charAt(val);

}

return plaintext;

}

public static void main(String[] args) {

Scanner sc =new Scanner(System.in);

String plaintext =sc.nextLine();

String encryptedText= doEncription(plaintext);

String decryptedText= doDecription(encryptedText);

System.out.println("Encriptedtext :"+encryptedText);

System.out.println("Decrypted text :"+decryptedText);

}

}

----------

Output

----------

Enter Plaintext : hello

Encriptedtext :mjnno

Decrypted text :ygcco

----------------------------------------------------------------------------------------------------------------

QUESTION 7 :

Implement DES/3DES/AES using Java APIs (use different modes + Base64 encoding)

a. Generate Keys

b. Store the Keys

c. Generate Initialization Vector (IV)

d. Select Mode (ECB/CBC/CFB)

e. Cipher Text → Base64 Encoding

----------------------------------------------------------------------------------------------------------------

----------------------------------------------------------------------------------------------------------------

i) AES

----------------------------------------------------------------------------------------------------------------

import javax.crypto.Cipher;

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.spec.SecretKeySpec;

import java.util.Base64;

import java.util.Scanner;

public class AESAlgorithm {

private static final String key = "aesEncryptionKey";

private static final String initVector = "encryptionIntVec";

public static String encrypt(String value) {

try {

IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));

SecretKeySpecskeySpec = new SecretKeySpec(key.getBytes("UTF-8"), "AES");

//Symmetric Encryption

Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");

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

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

return new String(Base64.getEncoder().encode(encrypted));

} catch (Exception ex) {

ex.printStackTrace();

}

return null;

}

public static String decrypt(String encrypted) {

try {

IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));

SecretKeySpecskeySpec = new SecretKeySpec(key.getBytes("UTF-8"), "AES");

Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");

cipher.init(Cipher.DECRYPT\_MODE, skeySpec, iv);

byte[] original = cipher.doFinal(Base64.getDecoder().decode(encrypted));

return new String(original);

} catch (Exception ex) {

ex.printStackTrace();

}

return null;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the String to Encrypt : ");

String originalString = scanner.nextLine();

String encryptedString = encrypt(originalString);

System.out.println("Encrypted String : " + encryptedString);

String decryptedString = decrypt(encryptedString);

System.out.println("Decrypted String : " + decryptedString);

}

}

-------------

output

-------------

Enter the String to Encrypt : hello

Encrypted String : rcEnllV7eBFiRdDF1WPlWg==

Decrypted String : hello

-------------

output

-------------

Enter the String to Encrypt : world

Encrypted String : vaai0vJVEbnIsDa5osO/vw==

Decrypted String : world

----------------------------------------------------------------------------------------------------------------

ii) DES

----------------------------------------------------------------------------------------------------------------

import javax.crypto.\*;

import javax.crypto.spec.DESKeySpec;

import java.nio.charset.StandardCharsets;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import java.util.Base64;

import java.util.Scanner;

public class DESAlgorithm {

public static void main(String[] args) throws NoSuchPaddingException, NoSuchAlgorithmException, IllegalBlockSizeException, BadPaddingException, InvalidKeyException, InvalidKeySpecException {

String keyString = "heloword";

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the Original String : ");

String originalText = scanner.nextLine();

DESKeySpecdesKeySpec = new DESKeySpec(keyString.getBytes());

SecretKeyFactorysecretKeyFactory = SecretKeyFactory.getInstance("DES");

SecretKeysecretKey = secretKeyFactory.generateSecret(desKeySpec);

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

String encryptedText = encryptText(originalText,cipher,secretKey);

System.out.println("Encrypted String: " + encryptedText);

String decryptedText = decryptText(encryptedText,cipher,secretKey);

System.out.println("Decrypted String: " + decryptedText);

}

public static String encryptText(String originalText,Ciphercipher,SecretKeysecretKey) throws BadPaddingException, IllegalBlockSizeException, InvalidKeyException {

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

byte[] originalByteArray = originalText.getBytes(StandardCharsets.UTF\_8);

byte[] encryptedByteArray = cipher.doFinal(originalByteArray);

byte[] encodedByteArray = Base64.getEncoder().encode(encryptedByteArray);

return new String(encodedByteArray);

}

public static String decryptText(String encryptedText,Ciphercipher,SecretKeysecretKey) throws InvalidKeyException, BadPaddingException, IllegalBlockSizeException {

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

byte[] encryptedByteArray = encryptedText.getBytes();

byte[] decodedBytes = Base64.getDecoder().decode(encryptedByteArray);

byte[] decryptedByteArray = cipher.doFinal(decodedBytes);

return new String(decryptedByteArray, StandardCharsets.UTF\_8);

}

}

------------

Output

------------

Enter the Original String : hello

Encrypted String: Xj5n+hQOoVM=

Decrypted String: hello

------------

Output

------------

Enter the Original String : world

Encrypted String: dQ+XuQgpIGA=

Decrypted String: world

----------------------------------------------------------------------------------------------------------------

QUESTION 8 :

Message Digest (MD5,SHA-1 hash, SHA-256 hash)

----------------------------------------------------------------------------------------------------------------

================================

1. MD5 (Message Digest)

================================

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class MD5Hash {

public static void main(String args[]) throws NoSuchAlgorithmException

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the String: ");

String string = scanner.nextLine();

System.out.println("Hash Value: " + convertToMD5(string));

}

public static String convertToMD5(String input)

{

try {

MessageDigestmDigest = MessageDigest.getInstance("MD5");

byte[] messageDigest = mDigest.digest(input.getBytes());

BigIntegerbigInteger = new BigInteger(1, messageDigest);

String hashValue = bigInteger.toString(16);

while (hashValue.length() < 32) {

hashValue = "0" + hashValue;

}

return hashValue;

}

catch (NoSuchAlgorithmException e) {

throw new RuntimeException(e);

}

}

}

-----------

output :

-----------

Enter the String: hello

Hash Value: 5d41402abc4b2a76b9719d911017c592

================================

1. SHA-1 hash

================================

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class SHA1Hash {

public static void main(String args[]) throws NoSuchAlgorithmException

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the String: ");

String string = scanner.nextLine();

System.out.println("Hash Value: " + convertToSHA(string));

}

public static String convertToSHA(String text) throws NoSuchAlgorithmException {

MessageDigestmessageDigest = MessageDigest.getInstance("SHA-1");

BigIntegerbigInteger = new BigInteger(1, messageDigest.digest(text.getBytes()));

String hexValue = bigInteger.toString(16);

while (hexValue.length() < 32) {

hexValue = "0" + hexValue;

}

return hexValue;

}

}

-----------

output:

----------

Enter the String: hello

Hash Value: aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d

======================================

1. SHA-256 hash

======================================

import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;

public class SHA256Hash {

public static void main(String args[]) throws NoSuchAlgorithmException

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the String: ");

String string = scanner.nextLine();

System.out.println("Hash Value: " + convertToSHA(string));

}

public static String convertToSHA(String text) throws NoSuchAlgorithmException {

MessageDigestmessageDigest = MessageDigest.getInstance("SHA-256");

BigIntegerbigInteger = new BigInteger(1, messageDigest.digest(text.getBytes()));

String hexValue = bigInteger.toString(16);

while (hexValue.length() < 32) {

hexValue = "0" + hexValue;

}

return hexValue;

}

}

-----------

output

-----------

Enter the String: hello

Hash Value: 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824

----------------------------------------------------------------------------------------------------------------

QUESTION 9 :

RSA

----------------------------------------------------------------------------------------------------------------

import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;

public class RSA

{

private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private Random r;

public RSA()

{

r = new Random();

p = BigInteger.probablePrime(bitlength, r);

q = BigInteger.probablePrime(bitlength, r);

N = p.multiply(q);

phi = p.subtract( BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

e = BigInteger.probablePrime(bitlength / 2, r);

while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 &&e.compareTo(phi) < 0)

{

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);

}

@SuppressWarnings("deprecation")

public static void main(String[] args) throws IOException

{

RSA rsa = new RSA();

DataInputStream in = new DataInputStream(System.in);

String teststring;

System.out.println("Enter the plain text:");

teststring = in.readLine();

System.out.println("Encrypting String: " + teststring);

System.out.println("String in Bytes: "

+ bytesToString(teststring.getBytes()));

// encrypt

byte[] encrypted = rsa.encrypt(teststring.getBytes());

// decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypting Bytes: " + bytesToString(decrypted));

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

}

private static String bytesToString(byte[] encrypted)

{

String test = "";

for (byte b : encrypted)

{

test += Byte.toString(b);

}

return test;

}

// Encrypt message

public byte[] encrypt(byte[] message)

{

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

// Decrypt message

public byte[] decrypt(byte[] message)

{

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}

----------------------------------------------------------------------------------------------------------------

QUESTION 10 :

RSA using API

----------------------------------------------------------------------------------------------------------------

import java.util.Scanner;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import java.security.\*;

import java.util.Base64;

public class RSA\_API{

public static void main(String[] args) throws NoSuchAlgorithmException,

NoSuchPaddingException, InvalidKeyException,

IllegalBlockSizeException, BadPaddingException {

KeyPairGeneratorkeyPairGenerator =

KeyPairGenerator.getInstance("RSA");

SecureRandomsecureRandom = new SecureRandom();

keyPairGenerator.initialize(2048,secureRandom);

KeyPair pair = keyPairGenerator.generateKeyPair();

PublicKeypublicKey = pair.getPublic();

String publicKeyString =

Base64.getEncoder().encodeToString(publicKey.getEncoded());

PrivateKeyprivateKey = pair.getPrivate();

String privateKeyString =

Base64.getEncoder().encodeToString(privateKey.getEncoded());

//Encrypt Hello world message

Scanner sc=new Scanner(System.in);

Cipher encryptionCipher = Cipher.getInstance("RSA");

encryptionCipher.init(Cipher.ENCRYPT\_MODE,privateKey);

System.out.print("Enter the plain text:");

String message = sc.nextLine();

byte[] encryptedMessage =

encryptionCipher.doFinal(message.getBytes());

String encryption =

Base64.getEncoder().encodeToString(encryptedMessage);

System.out.println("Encrypted message = "+encryption);

//Decrypt Hello world message

Cipher decryptionCipher = Cipher.getInstance("RSA");

decryptionCipher.init(Cipher.DECRYPT\_MODE,publicKey);

byte[] decryptedMessage =

decryptionCipher.doFinal(encryptedMessage);

String decryption = new String(decryptedMessage);

System.out.println("Decrypted message = "+decryption);

}

}

--------

output

--------

Enter the plain text:hello world

Encrypted message = NpBZl/hAQ8EGz51WrXBg0r754dzDylQtAWuUP/4YjvgNqPwHT/pZnSGSV19WNusDMJpeb76kfoOBS8i5YWHOZ+c/9qcpIBKEebEuYbEQq6WC67OUewT6gbiHEJctDZMSUD1jZ6a2bSZF1gHQxH35tkMj+wlN7vvyS3GX8pKI5YvHorpRMzJYHWkBeo6JocAe4XmSHvKyF976VNpgS1wkhTMximmcaiYIqY52yJXyESD3WWJMUYNvISGEHWchpwbTRkUZ7DNewDzkq2EC3mRFYWuUBjIMTt1JMRK2bHIX6Py3xxjiilaJOi4itUJYBE3+nLR0vI9BrqRdkDS9wD1NJA==

Decrypted message = hello world