**Practical.No-01**

**Aim : Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques.**

* **Caesar Cipher**
* **Monoalphabetic Cipher**
* **Rail Fence Cipher**
* **Simple Columnar Technique**
* **Vernam Cipher Source Code:**
  + **Caesar Cipher :**

import java.util.Scanner; public class CeasarCiphar {

String message; static int key;

static String encryptCaesar(String message1, int key1) { char ch;

String encryptMessage = "";

for (int i = 0; i < message1.length(); ++i) { ch = message1.charAt(i);

if (ch >= 'a' && ch <= 'z') { ch = (char) (ch + key1);

if (ch > 'z') {

ch = (char) (ch - 'z' + 'a' - 1);

}

encryptMessage += ch;

} else if (ch >= 'A' && ch <= 'Z') { ch = (char) (ch + key1);

if (ch > 'Z') {

ch = (char) (ch - 'Z' + 'A' - 1);

}

encryptMessage += ch;

} else {

encryptMessage += ch;

}

}

return encryptMessage;

}

static String descryptCeasar(String message1, int key1) { char ch;

String decryptMessage = "";

for (int i = 0; i < message1.length(); ++i) { ch = message1.charAt(i);

if (ch >= 'a' && ch <= 'z') {

ch = (char) (ch - key1);

if (ch < 'a') {

ch = (char) (ch + 'z' - 'a' + 1);

}

decryptMessage += ch;

} else if (ch >= 'A' && ch <= 'Z') {

ch = (char) (ch - key1);

if (ch > 'A') {

ch = (char) (ch + 'Z' - 'A' + 1);

}

decryptMessage += ch;

} else {

decryptMessage += ch;

}

}

return decryptMessage;

}

public static void main(String args[]) { String plainText;

int key;

String CipherText;

Scanner sc = new Scanner(System.in);

System.out.println("Enter a message to encrypt: "); plainText = sc.nextLine();

System.out.println("Enter key; "); key = sc.nextInt();

CipherText = encryptCaesar(plainText, key); System.out.println("Cipher Text = " + CipherText);

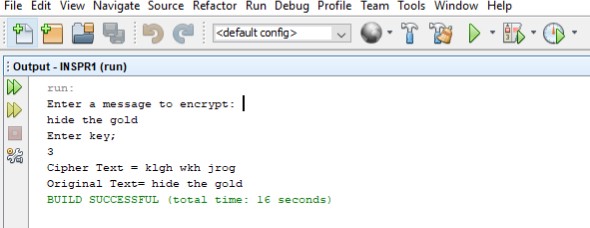
System.out.println("Original Text= " + descryptCeasar(CipherText,

key));

}

}

**Output:**



**Source Code:**

* + **Monoalphabetic Cipher:**

import java.util.Scanner;

public class MonoalplabetCipher {

public static void main(String args[]) { final char RALPHABETS[] = {'a', 'b', 'c',

'd','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v ','w','x','y','z'};

final char MALPHABETS[] =

{'Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K',' L','Z','X','C','V','B','B','N','M'};

Scanner s = new Scanner(System.in); String pltext;

char citext[] = new char[20]; char detext[] = new char[20];

int i, l;

System.out.print("Enter Plain Text: "); pltext = s.nextLine();

pltext = pltext.toLowerCase(); l = (pltext.length());

for (i = 0; i < l; i++) {

for (int j = 0; j < 26; j++) {

if (RALPHABETS[j] == pltext.charAt(i)) { citext[i] = MALPHABETS[j];

break;

}

}

}

System.out.print("Cipher Text: ");

for (i = 0; i < l; i++) { System.out.print(citext[i]);

}

String b = new String(citext); for (i = 0; i < l; i++) {

for (int j = 0; j < 26; j++) {

if (MALPHABETS[j] == b.charAt(i)) { detext[i] = RALPHABETS[j]; break;

}

}

}

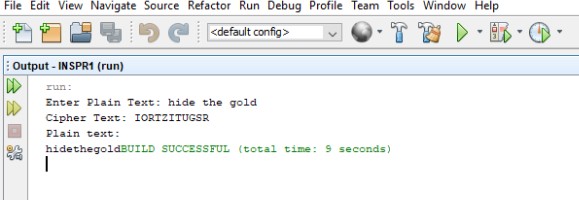
System.out.println("\nPlain text:");

for (i = 0; i < l; i++) { System.out.print(detext[i]);

}

}

}

**Output:**

**Source Code:**

* + **Rail Fence Cipher:** import java.util.\*; public class Railfence {

String Encryption(String plainText, int depth) throws Exception { int r = depth, len = plainText.length();

int c = len / depth; c = c + 1;

char mat[][] = new char[r][c]; int k = 0;

String cipherText = "";

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

for (int j = 0; j < r; j++) { if (k != len) {

mat[j][i] = plainText.charAt(k++); System.out.println("mat[" + j + "][" + i + "]=" +

mat[j][i]);

}

}

}

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

for (int j = 0; j < c; j++) { cipherText += mat[i][j];

}

}

return cipherText;

}

String Decryption(String cipherText, int depth) throws Exception { int r = depth, len = cipherText.length();

int c = len / depth;

char mat[][] = new char[r][c]; int k = 0;

String plainText = "";

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

for (int j = 0; j < c; j++) { mat[i][j] = cipherText.charAt(k++);

}

}

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

for (int i = 0; i < r; i++) { plainText += mat[i][j];

}

}

return plainText;

}

}

class RailFenceB {

public static void main(String args[]) throws Exception { Scanner sc = new Scanner(System.in);

int depth;

String plainText, cipherText, decryptedText;

System.out.println("Enter plain Text"); plainText = sc.nextLine();

System.out.println("Enter Depth(No of Rails) for Encryotion: "); depth = sc.nextInt();

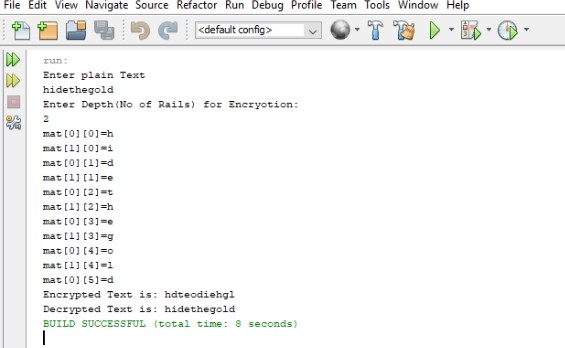
Railfence rf = new Railfence();

cipherText = rf.Encryption(plainText, depth); System.out.println("Encrypted Text is: " + cipherText);

decryptedText = rf.Decryption(cipherText, depth); System.out.println("Decrypted Text is: " + decryptedText);

}

}

**Output:**

**Source Code:**

* + **Simple Columnar Technique:**

import java.io.\*;

import java.util.Scanner; public class SCT {

public static void main(String args[]) throws Exception { Scanner sc = new Scanner(System.in); System.out.println("Enter your plain text: "); String accept = sc.nextLine();

System.out.println("Enter the no. of rows"); int r = Integer.parseInt(sc.nextLine()); System.out.println("Enter the no. of cols"); int c = Integer.parseInt(sc.nextLine());

int count = 0;

char cont[][] = new char[r][c];

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

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

if (count >= accept.length()) { cont[i][j] = ' ';

} else {

cont[i][j] = accept.charAt(count); count++;

}

}

}

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

for (int j = 0; j < c; j++) { System.out.print("\t" + cont[i][j]);

}

System.out.print("\n");

}

System.out.println("\nEnter the order of cols you want to view them in");

int choice[] = new int[c]; for (int k = 0; k < c; k++) {

System.out.println("Choice " + k + " -> "); choice[k] = Integer.parseInt(sc.nextLine());

}

System.out.println("\ncipher text in matrix is -> "); String cipher = "";

for (int j = 0; j < c; j++) { int k = choice[j];

for (int i = 0; i < r; i++) { cipher += cont[i][k];

}

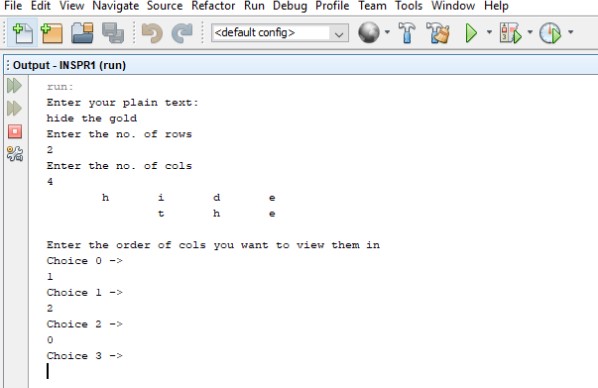
}

System.out.println(cipher);

}

}

**Output:**



**Source Code:**

* + **Vernam Cipher:** import java.lang.Math; import java.util.Scanner;

public class Vernam {

public static void main(String args[]) { Scanner sc = new Scanner(System.in); String plainText = null, key = null;

System.out.println("Enter plain Text"); plainText = sc.nextLine();

char[] arText = plainText.toCharArray();

System.out.println("Enter the Key "); key = sc.nextLine();

char[] arKey = key.toCharArray();

char[] cipherText = new char[13]; System.out.println("Encoded " + plainText + " to be ");

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

cipherText[i] = (char) (arText[i] ^ arKey[i]); System.out.print(cipherText[i]);

}

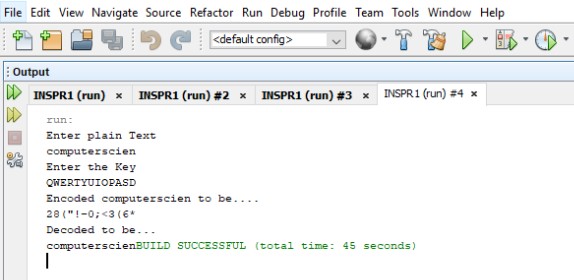
System.out.println("\nDecoded to be..."); for (int i = 0; i < cipherText.length; i++) {

char temp = (char) (cipherText[i] ^ arKey[i]); System.out.print(temp);

}

}

}

**Output:**

**Practical.No:02**

**Aim : Implement the RSA algorithm for public-key encryption and decryption, and explore its properties and security considerations.**

**Source Code:**

* + **RSA:**

import java.math.\*; import java.util.\*;

public class RSA {

public static void main(String args[]){ int p,q,n,z,d=0,e,i;

double c; BigInteger msgback;

p=5; q=11;

int msg=12;

n=p\*q;

z=(p-1)\*(q-1);

System.out.println("The value of Z ="+z);

for(e=2;e<z;e++){ if(gcd(e,z)==1){

break;

}

}

System.out.println("The value of e ="+e); for(i=0;i<=9;i++){

int x=1+(i\*z); if(x%e==0){

d=x/e; break;

}

}

System.out.println("The value of e="+e); for(i=0;i<=9;i++){

int x=1+(i\*z);

if(x%e==0){

d=x/e; break;

}

}

System.out.println("The value of d="+d); c=(Math.pow(msg,e))%n; System.out.println("Encrypted message is:"+c);

BigInteger N= BigInteger.valueOf(n);

BigInteger C=BigDecimal.valueOf(c).toBigInteger(); msgback=(C.pow(d)).mod(N); System.out.println("Decrypted message is:"+msgback);

}

static int gcd(int e, int z){ if(e==0){

return z;

} else{

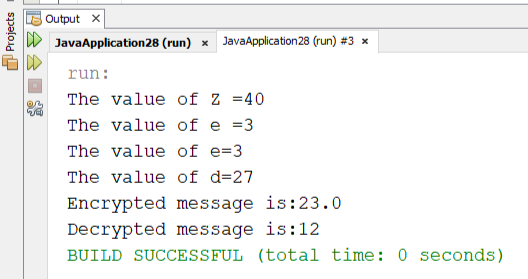
return gcd(z%e,e);

}

}

}

**Output:**



**Practical.No:03**

**Aim : Implement algorithms to generate and verify message authentication codes (MACs) for ensuring data integrity and authenticity.**

**Source Code:**

* + **Message Authentication Codes:**

import java.math.BigInteger;

import java.nio.charset.StandardCharsets; import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException; public class MDS {

public static String toHexString(byte[] hash){ BigInteger number = new BigInteger(1,hash);

StringBuilder hexString=new StringBuilder(number.toString(16));

while(hexString.length () > 32){ hexString.insert(0,'0');

}

return hexString.toString();

}

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

{

try{

System.out.println("HashCode Generated by MD5 for:");

String s1=" Information and Security"; MessageDigest md;

md = MessageDigest.getInstance("MD5");

byte[] hash=md.digest(s1.getBytes(StandardCharsets.UTF\_8));

System.out.println("Message Digest : "+s1+" : " +

toHexString(hash));

}

catch(NoSuchAlgorithmException e){

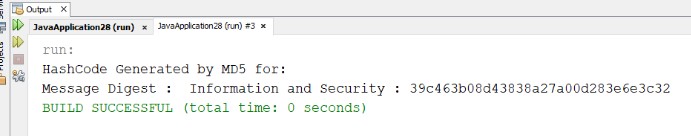
System.out.println("Exception throw for incorrect algorithm :

"+e);

}

}

}

**Output:**

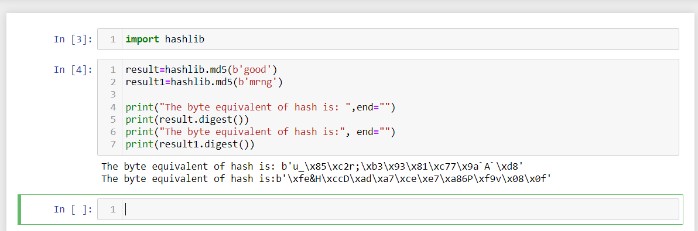
**1)** In Jupiter notebook:MD5

import hashlib result=hashlib.md5(b'good') result1=hashlib.md5(b'mrng')

print("The byte equivalent of hash is: ",end="") print(result.digest())

print("The byte equivalent of hash is:", end="") print(result1.digest())

**Output:**



**2)**

#python code to implement MD5 import hashlib

#provide Input and apply MD5 result=hashlib.md5(b'good')

#convert hsh value into hexadecimal result=result.digest()

#Display Result print('Message Digest',result)



SHA

1)in jupiternotebook

#python code to implement SHA import hashlib

str=input('Enter string to encode')

#Apply SHA1 result=hashlib.sha1(str.encode())

#convert it into hexadecimal value result=result.hexdigest()

#Display Result

print('Output of SHA1',result)



**Practical.No:04**

**Aim : Implement digital signature algorithms such as RSA-based signatures, and verify the integrity and authenticity of digitally signed messages.**

**Source Code:**

* **Digital Signatures:** package digital\_signature; import java.security.PrivateKey; import java.security.\*;

import java.util.Scanner;

import javax.xml.bind.DatatypeConverter;

public class Digital\_Signature {

private static final String SIGNING\_ALGORITHM="SHA256withRSA"; private static final String RSA="RSA";

private static Scanner sc;

// Function to implement Digital signature

//using SHA256 and RSA algorithm

//by passing private key.

public static byte[] Create\_Digital\_Signature(byte[] input, PrivateKey Key)throws Exception{ Signature signature=Signature.getInstance(SIGNING\_ALGORITHM); signature.initSign(Key);

signature.update(input); return signature.sign();

}

public static KeyPair Generate\_RSA\_Keypair() throws Exception{ SecureRandom secureRandom = new SecureRandom(); KeyPairGenerator keyPairGenerator=KeyPairGenerator.getInstance(RSA); keyPairGenerator.initialize(2048,secureRandom);

return keyPairGenerator.generateKeyPair();

}

public static boolean Verify\_Digital\_Signature(byte[] input,byte[] signatureToVerify,PublicKey key) throws Exception{

Signature signature= Signature.getInstance(SIGNING\_ALGORITHM); signature.initVerify(key);

signature.update(input);

return signature.verify(signatureToVerify);

}

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

String input="Good Morning";

KeyPair keyPair=Generate\_RSA\_Keypair();

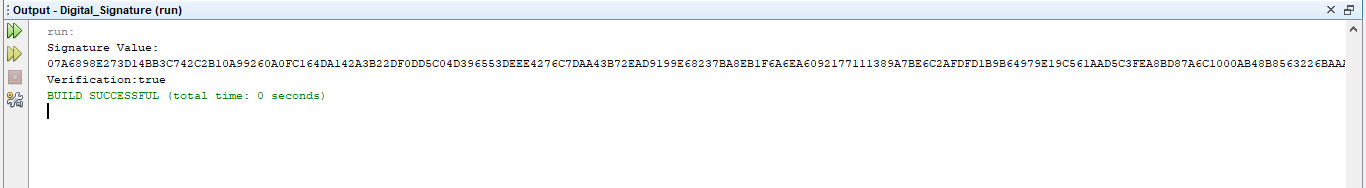
byte[] signature=Create\_Digital\_Signature(input.getBytes(),keyPair.getPrivate()); System.out.println("Signature Value:\n"+DatatypeConverter.printHexBinary(signature));

System.out.println("Verification:"+Verify\_Digital\_Signature(input.getBytes(),signature,keyPair.getPu blic()));

}

}

**Output:**



**Practical.No:05**

**Aim: Implement the Diffie-Hellman key exchange algorithm to securely exchange keys between two entities over an insecure network.**

**Source Code:**

* **DH:**

import java.util.\*; public class DH {

public static void main(String args[]){ Scanner sc=new Scanner(System.in);

System.out.println("Enter PRIME NUMBER 1 p:"); int p =sc.nextInt();

System.out.println("Enter PRIME NUMBER 2 g:"); int g=sc.nextInt();

System.out.println("Choose 1st secret no(Alice)'a':"); int a= sc.nextInt();

System.out.println("choose 2nd secret no(Bob)'b':"); int b=sc.nextInt();

int A=(int)Math.pow(g,a)%p; int B=(int)Math.pow(g, p)%p;

System.out.println("public key of alice:"+A); System.out.println("Public key of Bob:"+B);

int S\_A = (int)Math.pow(B, a)% p; int S\_B = (int)Math.pow(A,b)% p;

System.out.println("Shared key of alice S\_A:"+S\_A); System.out.println("Shared key of Bob S\_B:"+S\_B);

if(S\_A == S\_B){

System.out.println("Alice and bob can communicate"+"with each other!!!");

System.out.println("They share a secret no=" + S\_A);

}

else{

System.out.println("Alice and Bob cannot"+"communicate with each other!!!" );

}

}

}

**Output:**

