1.Caeser Cipher:

import java.util.Scanner;

public class CaesarCipher {

public static String encode(String enc, int offset){

StringBuffer encoded = new StringBuffer();

for (char i : enc.toCharArray()){

if (Character.isLetter(i)){

if (Character.isUpperCase(i)){

encoded.append((char) ('A'+(i-'A'+ offset)% 26));

}

else {

encoded.append((char) ('a'+(i-'a'+ offset)% 26));

}

}

else{

encoded.append(i);

}

}

return encoded.toString();

}

public static String decode(String enc,int offset) {

return encode(enc, 26 - offset);

}

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

Scanner scan = new Scanner(System.in);

System.out.println(" Simulating Caesar Cipher\n");

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

String msg = scan.next();

System.out.println("Enter the key");

Integer key = scan.nextInt();

String cipher=encode(msg, key);

System.out.println("Encrypted Message:");

System.out.println(cipher);

System.out.println("Decrypted Message:");

System.out.println(decode(cipher, key));

}

}

2. Playfair Cipher:

import java.awt.Point;

import java.util.Scanner;

public class PlayFairCipher{

private static char[][] charTable;

private static Point[] positions;

private static String prepareText(String s) {

s = s.toUpperCase().replaceAll("[^ A-Z]", "");

return s.replace("J","I");

}

private static void createTbl(String key) {

charTable = new char[5][5];

positions = new Point[26];

String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ");

int len = s.length();

for (int i = 0, k = 0; i < len; i++) {

char c = s.charAt(i);

if (positions[c - 'A'] == null) {

charTable[k / 5][k % 5] = c;

positions[c - 'A'] = new Point(k / 5, k % 5);

k++;

}

}

System.out.println("The key square");

for (int i = 0; i < 5; i++) {

for (int j = 0; j < 5; j++) {

System.out.print(charTable[i][j] + " ");

}

System.out.println();

}

}

private static String codec(StringBuilder txt, int dir) {

int len = txt.length();

for (int i=0;i< len;i += 2) {

char a = txt.charAt(i);

char b = txt.charAt(i + 1);

int row1 =positions[a - 'A'].x;

int row2= positions[b - 'A'].x;

int col1 =positions[a - 'A'].y;

int col2 =positions[b - 'A'].y;

if (row1 == row2) {

col1 = (col1 + dir) % 5;

col2 = (col2 + dir) % 5;

}

else if (col1 == col2) {

row1 = (row1 + dir) % 5;

row2 = (row2 + dir) % 5;

}

else {

int tmp = col1;

col1 = col2;

col2 =tmp;

}

txt.setCharAt(i, charTable[row1][col1]);

txt.setCharAt(i + 1, charTable[row2][col2]);

}

return txt. toString();

}

private static String encode(String s) {

StringBuilder sb = new StringBuilder(s);

for (int i =0; i< sb.length(); i += 2){

if (i == sb.length()-1){

sb.append(sb.length()% 2==1? 'Z': "");

}

else if (sb.charAt(i) == sb.charAt(i + 1)) {

sb.insert(i + 1, 'Z');

}

}

return codec(sb, 1);

}

private static String decode(String s) {

return codec(new StringBuilder(s), 4);

}

public static void main(String[] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter the key");

String key = scan.next();

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

String txt = scan.next();

createTbl(key);

String enc = encode(prepareText(txt));

System.out.println("Simulating Playfair Cipher \n-- ");

System.out.println("Input Message:"+ txt);

System.out.println("Encrypted Message: " + enc);

System.out.println(" Decrypted Message: "+ decode(enc));

}

}

3. Hill Cipher:

import java. util. Scanner;

public class HillCipher {

public static int[][] keymat = new int[3][3];

private static String encode(char a, char b, char c) {

String ret= "";

int x, y, z;

int posa = (int) a - 65;

int posb=(int) b - 65;

int posc= (int) c - 65;

x= posa \* keymat[0][0]+ posb \* keymat[1][0] + posc \* keymat[2][0];

y= posa \* keymat[0][1]+ posb \* keymat[1][1] + posc \* keymat[2][1];

z= posa \* keymat[0][2]+ posb \* keymat[1][2] + posc \* keymat[2][2];

a=(char)(x % 26 + 65);

b=(char)(y % 26 + 65);

c=(char)(z % 26 + 65);

ret = ""+a+b+c;

return ret;

}

public static void main(String[] args) {

String msg ;

String enc = "";

int n;

System.out.println("Simulation of Hill Cipher\n------- ");

System.out.println ("Enter the 3x3 key matrix(0 to 25)");

Scanner scan=new Scanner(System.in);

for(int i=0; i<3;i++){

for (int j=0; j < 3 ;j++){

keymat[i][j] = scan.nextInt();

}

}

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

msg=scan.next();

msg+=scan.nextLine();

msg= msg. toUpperCase();

msg= msg.replaceAll("\\s", "");

n = msg.length() % 3;

if (n!=0) {

for (int i = 1; i <= (3 - n); i++) {

msg += 'X';

}

}

System.out.println("Padded plain text: " + msg);

char[] plaintext = msg.toCharArray();

for (int i = 0; i< msg.length(); i += 3){

enc += encode(plaintext[i], plaintext[i + 1], plaintext[i + 2]);

}

System.out.println("Encoded cipher text " + enc);

}

}

4.Vigenere Cipher:

import java.util.Scanner;

public class VigenereCipher {

static String encode(String ptext, String key) {

String ctext = "";

ptext = ptext.toUpperCase();

for (int i =0,j = 0; i< ptext.length(); i++) {

char c=ptext.charAt(i);

if (c<'A'||c> 'Z'){

continue;

}

ctext += (char) ((c + key.charAt(j)-2\*'A') % 26+ 'A');

j=++j % key.length();

}

return ctext;

}

static String decode(String ctext, String key) {

String ptext = "";

ctext=ctext.toUpperCase();

for (int i = 0, j = 0; i < ctext.length(); i++) {

char c=ctext.charAt(i);

if (c < 'A' || c > 'Z') {

continue;

}

ptext += (char) ((c - key.charAt(j) + 26) % 26 + 'A');

j = ++j % key.length();

}

return ptext;

}

public static void main (String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("'Simulating Vigenere Cipher\n-----");

System.out.println("Enter the key");

String key= scan.next();

key=key.toUpperCase();

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

String msg = scan.next(); msg+=scan.nextLine ();

String cipher = encode(msg, key);

System.out.println("Encrypted cipher text\n" + cipher);

String plain =decode(cipher, key);

System.out.println("Decrypted plain text\n" + plain);

}

}

5. RailFence Cipher:

import java.util.Scanner;

public class railfenceCipher {

String encode(String msg) {

int len = msg.length();

int k = 0, i = 0;

String enc="";

while (k!=len) {

enc += msg.charAt(i);

i += 2;

if (i >= len) {

i=1;

}

k++;

}

return enc;

}

String decode(String encmsg) {

int len = encmsg.length();

int k = 0, i = 0;

int j = (int) Math.ceil((double) len / 2);

String dec = "";

while (k != len) {

if (k % 2 == 0) {

dec += encmsg.charAt(i);

i++;

}

else{

dec += encmsg.charAt(j);

j++;

}

k++;

}

return dec;

}

public static void main(String[] args) {

railfenceCipher rf = new railfenceCipher();

String msg, enc, dec;

Scanner scan = new Scanner(System.in);

System.out.println("Simulating Railfence Cipher\n -- ");

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

msg = scan.next();

msg += scan.nextLine();

enc = rf.encode(msg);

dec = rf.decode(enc);

System.out.println("Encrypted Message: " + enc);

System.out.println("Decrypted Message: " + dec);

}

}

6. RowColumn:

import java.util.\*;

public class RowColumn {

public static void main(String[] args ){

Scanner sc = new Scanner(System.in);

System.out.println("SIMULATING ROW & COLUMN TRANSPOSITION CIPHER");

System.out.println("-----------------------");

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

String pl= sc.next();

pl+=sc.nextLine();

System.out.print("Enter the number of columns: ");

int col = sc.nextInt();

String s= "";

s= pl.replaceAll(" ","");

int len=s.length();

int k = 0;

int row= (int)Math.ceil((double)len/col);

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

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

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

if (k < len) {

ch[i][j] = s.charAt(k);

k++;

}

else {

ch[i][j]= 'X';

}

}

}

System.out.println("The row and column matrix");

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

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

System.out.print(ch[i][i] + " ");

}

System.out.println();

}

System.out.print("The cipher text is: ");

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

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

System.out.print(ch[j][i]);

}

}

}

}

7. DES:

import java.util.Scanner;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

public class DES {

public static void main(String[] argv) {

try {

System.out.println("Message Encryption Using DES Algorithm\n ");

KeyGenerator keygenerator = KeyGenerator.getInstance("DES");

SecretKey myDesKey = keygenerator.generateKey();

Cipher desCipher;

desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

desCipher.init(Cipher.ENCRYPT\_MODE, myDesKey);

Scanner scan = new Scanner(System.in);

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

String ptext = scan.next();

ptext += scan.nextLine();

byte[] text = ptext.getBytes();

System.out.println("Plain text [Byte Format]: " + text);

byte[] textEncrypted = desCipher.doFinal(text);

System.out.println("Encrypted plain text:" + textEncrypted);

desCipher.init(Cipher.DECRYPT\_MODE, myDesKey);

byte[] textDecrypted = desCipher.doFinal(textEncrypted);

System.out.println("Decrypted cipher text: " + new String(textDecrypted));

} catch (Exception e) {

e.printStackTrace();

}

}

}

8.AES:

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Arrays;

import java.util.Base64;

import java.util.Scanner;

public class AES {

private static SecretKeySpec secretKey;

public static void setKey(String myKey) {

MessageDigest sha = null;

try {

byte[] key = myKey.getBytes(StandardCharsets.UTF\_8);

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

key = sha.digest(key);

key = Arrays.copyOf(key, 16);

secretKey = new SecretKeySpec(key, "AES");

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

}

public static String encrypt(String strToEncrypt,String secret){

try{

setKey(secret);

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

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

return Base64. getEncoder().encodeToString(cipher.doFinal (strToEncrypt.getBytes(StandardCharsets.UTF\_8)));

}

catch (Exception e){

System.out.println("Error while encrypting:" + e);

}

return null;

}

public static String decrypt(String strToDecrypt, String secret){

try {

setKey(secret);

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

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

return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));

} catch (Exception e) {

System.out.println("Error while decrypting: " + e);

}

return null;

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

System.out.println("Encryption Using AES Algorithm\n-- ") ;

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

String secretKey= scan.next();

secretKey += scan.nextLine();

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

String originalString=scan.next();

originalString += scan.nextLine();

String encryptedString= encrypt(originalString, secretKey);

String decryptedString= decrypt(encryptedString, secretKey);

System.out.println("Original plain text: " + originalString);

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

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

}

}

9.RSA:

<html>

<head>

<title>RSA Encryption</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0 ">

</head>

<body>

<center>

<h1>RSA Algorithm</h1>

<h2>Implemented Using HTML & Javascript</h2>

<hr>

<table>

<tr>

<td>Enter First Prime Number (p):</td>

<td><input type="number" id="p"></td>

</tr>

<tr>

<td>Enter Second Prime Number (q):</td>

<td><p ><input type="number" id="q"></p></td>

</tr>

<tr>

<td>Enter the Message (cipher text):<br>[A=1, B=2,.]</td>

<td><input type="number" id="msg"></td>

</tr>

<tr>

<td>Public Key (e,n):</td>

<td> <p id="publickey"></p> </td>

</tr>

<tr>

<td>Private Key (d,n):</td>

<td> <p id="privatekey"></p></td>

</tr>

<tr>

<td>Cipher Text</td>

<td><p id="ciphertext"></p></td>

</tr>

<tr>

<td><button onclick="RSA();">Apply RSA</button></td>

</tr>

</table>

</center>

</body>

<script type="text/javascript">

function checkPrime(n){

var i,flag=true;

n=parseInt(n);

for(i=2;i<=n-1;i++){

if(n%i==0){

flag=false;

break;

}

return flag

}

function RSA(){

var gcd, p, q, M, n, t, e, i, x, C, d, dt;

gcd function (a, b) {return (b) ? a: ged(b, a % b); };

p=document.getElementById('p').value

q=document.getElementById('q').value;

if((checkPrime(p) && checkPrime(q)){alert("p &q should always be prime numbers");

return;

}

M=document.getElementById('msg').value;

n=p\*q;

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

for (e = 2; e<t;et+) {

if (gcd(e, t)== 1){

break;

}}

i=0;

while(1){

x=1+i\*t;

if(x%e==0){

d=x/e;

break;

}

i++;

}

C= (Math.pow(M, e).toFixed(0)) % n;

document.getElementById('publickey').innerHTML =e+","+ n;

document.getElementById('privatekey').innerHTML =d+","+n;

document.getElementById('ciphertext').innerHTML = C;

}

</script>

</html>

10.DHKE:

import java.util.Scanner;

class DHKE{

public static void main (String[] args) {

int q= 23;

int a =5;

System.out.println("Simulation of Diffie-Hellman key exchange algorithm\n-----");

System.out.println("Global public element prime number q= "+ q);

System.out.println("Global public element a =" + a);

Scanner scan = new Scanner(System.in);

System.out.print("Enter the private key of user A (Xa<q Here the value of q is 23): ");

int Xa = scan.nextInt();

System.out.print("Enter the private key of user B (Xb<q Here the value of qis 23): ");

int Xb= scan.nextInt();

double Ya = (Math.pow(a, Xa)) % q;

double Kb = (Math.pow(Ya, Xb)) % q;

double Yb = (Math.pow(a, Xb)) % q; double Ka = (Math.pow(Yb, Xa)) % q;

System.out.println("Public key of User A (Ya): "+ Ya);

System.out.println("User B computes signature using public key of user A (Kb):"+ Kb);

System.out.println("Public key of User B (Yb): " + Yb);

System.out.println("User A computes signature using public key of user B (Ka): "+ Ka);

if (Ka == Kb)

System.out.println("'Success: Shared key matches! "+ Ka);

else

System.out.println("Error: Shared key does not match");

}

}

11. SHA1:

import java.security.\*;

import java.util.Scanner;

public class SHA1 {

public static void main(String[] a){

try{

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

System.out.println("Message digest object info: \n--");

System.out.println("Algorithm-"+ md.getAlgorithm());

System.out.println("Provider=" + md.getProvider());

System.out.println("ToString"+ md.toString());

Scanner scan = new Scanner(System.in);

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

String input = scan.next() ;

input += scan.nextLine();

md.update(input.getBytes());

md.update(input.getBytes());

byte[] output= md.digest();

System.out.println("SHA1(\""+ input + "\")="+ bytesTo Hex(output);

catch (Exception e){

System.out.println("Exception:" + e)

}

}

private static String bytesToHex(bytel b){

char hexDigit[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'}

StringBuffer buf = new StringBuffer();

for (byte aB : b) {

buf.append(hexDigit[(aB >> 4) & Ox0f]);

buf.append(hexDigit[aB & 0xOf]);

}

return buf.toString();

}

}