**PASSWORD EXTRACTION:**

import hashlib

ascii\_hash = dict()

for i in range(-1,26):

for j in range(-1,26):

for k in range(26):

if (i==-1):

if (j==-1):

hash\_value = hashlib.md5(chr(65+k).encode())

ascii\_hash[chr(65+k)] = str(hash\_value.hexdigest())

continue

hash\_value = hashlib.md5((chr(65+j)+chr(65+k)).encode())

ascii\_hash[chr(65+j)+chr(65+k)] = str(hash\_value.hexdigest())

continue

hash\_value = hashlib.md5((chr(97+i)+chr(65+j)+chr(65+k)).encode())

ascii\_hash[chr(65+i)+chr(65+j)+chr(65+k)] = str(hash\_value.hexdigest())

key\_list = list(ascii\_hash.keys())

val\_list = list(ascii\_hash.values())

# print(val\_list)

while(True):

input\_text = input("\nEnter the text(-1 to exit): ")

if (input\_text=="-1"):

break

input\_text = input\_text.upper()

print(input\_text)

output\_hash = ascii\_hash[input\_text]

# print(ascii\_hash)

print("Output Hash: ",output\_hash)

print("Generating password from the hash...")

print("Password:",key\_list[val\_list.index(output\_hash)])

# for i in ascii\_hash:

# if (ascii\_hash[i] == output\_hash):

# print("Password: ",i)

**SHA:**

import java.util.Collections;

import java.util.Scanner;

import java.lang.String;

public class Main{

public static void leftshift(long val,int count)

{

String x = Long.toBinaryString(val);

if (x.length()<32) {

x = String.join("", Collections.nCopies(32-x.length(),"0"))+x;

}

x = x.substring(count)+x.substring(0,count);

String x1 = "";

for (int i=0;i<x.length();i=i+4) {

x1 += Integer.toHexString(Integer.parseInt(x.substring(i,i+4),2));

}

System.out.printf("\n shift of %h : %h \n",val,Long.parseLong(x1,16));

}

public static void main(String args[]){

long a = 0x67452301L;

long b = 0xabcdef12L;

long c = 0xa1b2c3e4L;

long d = 0x87654321L;

long e = 0x45ab67cdL;

long f = 0x12abcdefL; z

int count;

Scanner ip= new Scanner(System.in);

System.out.print("enter the shift : ");

count= ip.nextInt();

leftshift(a,count);

leftshift(b,count);

leftshift(c,count);

leftshift(d,count);

leftshift(e,count);

leftshift(f,count);

long res = ((b&c)|((~b)&d));

System.out.printf("\n expression : %h \n",res);

long xor = a^b;

System.out.printf("\n xor(a and b) : %h \n",xor);

}

}

**PRIMITIVE ROOTS:**

import java.math.\*;

import java.util.\*;

public class primRoot {

public static void main(String arg[]){

Scanner scan = new Scanner(System.in);

System.out.print("Number:");

BigInteger p = new BigInteger(scan.nextLine());

BigInteger pi\_p = p.subtract(BigInteger.ONE);

BigInteger arr[] = new BigInteger[p.intValue()];

int iter = 0;

for (BigInteger i = new BigInteger("2");i.compareTo(pi\_p)<0;i=i.add(BigInteger.ONE)) {

if (i.isProbablePrime(1) && pi\_p.mod(i).compareTo(BigInteger.ZERO)==0) {

arr[iter++] = pi\_p.divide(i);

}

}

System.out.println("Working");

for (BigInteger i = new BigInteger("2");i.compareTo(p)<0;i=i.add(BigInteger.ONE)) {

int flag = 0;

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

if ((i.modPow(arr[j],p)).compareTo(new BigInteger("1"))==0) {

flag = 1;

break;

}

}

if (flag==0) {

System.out.print(i+", ");

}

}

}

}

**AES:**

import java.util.\*;

public class modAES{

static int key[][] = new int[4][44];

static int mainkey[][] = {

{0x2b, 0x28, 0xab, 0x9},

{0x7e, 0xae, 0xf7, 0xcf},

{0x15, 0xd2, 0x15, 0x4f},

{0x16, 0xa6, 0x88, 0x3c}

};

static final int[][] sbox = {

{0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76},

{0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0},

{0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15},

{0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75},

{0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84},

{0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf},

{0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8},

{0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2},

{0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73},

{0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb},

{0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79},

{0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08},

{0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a},

{0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e},

{0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf},

{0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16}

};

static void display(){

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

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

System.out.print(Integer.toHexString(key[i][j]) + " ");

}

System.out.println();

}

}

static void quadKey(int c){

int[] shift = new int[4];

int[] sboxer = new int[4];

int count=0;

int c1=0;

for (int i=0; i<4; i++){

String split;

if (i == 3){

split = Integer.toHexString(key[0][c-1]);

shift[count++] = key[0][c-1];

}

else{

split = Integer.toHexString(key[i+1][c-1]);

shift[count++]=key[i+1][c-1];

}

if (split.length() == 1){

split = "0" + split.charAt(0);

}

int row = Integer.parseInt(split.charAt(0) + "", 16);

int col = Integer.parseInt(split.charAt(1) + "", 16);

sboxer[c1++]=sbox[row][col];

}

System.out.println("Top circular shift: ");

for (int n=0;n<4;n++){

System.out.println(Integer.toHexString(shift[n])+"\t");

}

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

System.out.println("Substitute bytes: ");

for (int m=0;m<4;m++){

System.out.println(Integer.toHexString(sboxer[m])+"\t");

}

}

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

for (int j=0; j<mainkey[0].length; j++){

key[i][j] = mainkey[i][j];

}

}

System.out.println("Main Key: ");

display();

int count = 1;

quadKey(4);

}

}

**FREQUENCY VALUE ANALYSIS:**

alpha = "abcdefghijklmnopqrstuvwxyz"

def decrypt(word, key):

plain = ""

for i in word:

plain += alpha[abs(alpha.index(i) - key)]

return plain

cipher = input("Enter cipher text: ")

d = dict()

for i in cipher:

d[i] = d.get(i, 0) + 1

print(max(d.values()))

maxer = ""

for i in d:

if d[i] == max(d.values()):

maxer = i

break

print("maximum occurence is " + maxer + " with count " + str(max(d.values())))

while True:

inp = input("Enter letter to be replaced with: ")

key = abs(alpha.index(inp) - alpha.index(maxer))

plain = decrypt(cipher, key)

print(plain)

choice = int(input("1.continue 2.break"))

if choice == 2:

break

**RSA:**

package lab\_rec;

import java.util.\*;

public class newrsa{

int multiInv(int e, int phi){

int t, t1=0, t2=1, r=0, q=0, a=phi, b=e;

while(b!=0){

r=a%b;

q=a/b;

t=t1-(q\*t2);

t1=t2;

t2=t;

a=b;

b=r;

}

if(t1>=0)

return t1;

else

return t1+phi;

}

int generateD(int e, int phi){

int d = multiInv(e, phi);

return d;

}

public static void main(String[] args){

newrsa obj = new newrsa();

int num[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97};

int n = 159, p=0, q=0, e=3;

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

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

int prd = num[j]\*num[i];

if(prd==n){

p = num[j];

q = num[i];

break;

}

}

}

System.out.println("P: "+p);

System.out.println("Q: "+q);

int phi = (p-1)\*(q-1);

System.out.println("Phi: "+phi);

int d = obj.generateD(e, phi);

System.out.println("D: "+d);

}

}

**HILL CIPHER:**

import java.util.Locale;

import java.util.Scanner;

public class Main {

static int det;

public static void main(String[] args) {

Scanner ip = new Scanner(System.in);

System.out.print("Enter a word: ");

String word = ip.nextLine().toLowerCase();

if (word.length()%9!=0) {

word += "x".repeat(9-word.length()%9);

}

String inword = word;

int arr[][] = numEncoding(word);

display(arr);

int key[][] = new int[3][3];

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

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

System.out.printf("Enter the value of key[%d][%d]: ",i+1,j+1);

key[i][j] = Integer.parseInt(ip.nextLine());

}

}

arr = matMultiplication(arr,key);

display(arr);

String outword = charEncoding(arr);

System.out.println("Encrypted Word: "+outword);

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

word = ip.nextLine();

arr = numEncoding(word);

int cofac[][] = cofactor(key);

System.out.println("Cofactor Matrix: ");

display(cofac);

int adjoint[][] = adjoint(cofac);

System.out.println("Adjoint Matrix: ");

display(adjoint);

arr = matMultiplication(arr,adjoint);

det = inv(det);

System.out.println("Determinant: "+det);

if (det!=-1) {

arr = scalarMultiplication(arr,det);

display(arr);

}

word = charEncoding(arr);

System.out.println("Decrypted Word: "+word);

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

inword = ip.nextLine();

System.out.print("Enter Encrypted word: ");

word = ip.nextLine();

int arr2[][] = numEncoding(inword);

arr = numEncoding(word);

display(arr2);

display(arr);

int key1[][] = scalarMultiplication(matMultiplication(adjoint(cofactor(arr2)),arr),inv(det));

System.out.println("Key Inverse: ");

display(key1);

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

display(scalarMultiplication(adjoint(cofactor(key1)),inv(det)));

}

public static int[][] scalarMultiplication(int[][] arr, int n) {

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

for (int j = 0; j < arr[0].length; j++) {

arr[i][j] = (n\*arr[i][j])%26;

if (arr[i][j] <0) arr[i][j] += 26;

}

}

return arr;

}

public static int inv(int det) {

int a=26,b=det,t1=0,t2=1,t=0,q=0,r=0;

while (b!=0) {

r = a%b;

q = a/b;

a = b;

b = r;

t = t1-(q\*t2);

t1 = t2;

t2 = t;

}

if (a==1)

return t1;

else

return -1;

}

public static int[][] cofactor(int[][] key) {

int[][] cofactor = new int[key.length][key[0].length];

det = 0;

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

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

int sum1[] = new int[4],i1=0;

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

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

if (k==i || l==j) continue;

sum1[i1++] = key[k][l];

}

}

cofactor[i][j] = (((int)Math.pow(-1,i+j)\*((sum1[0]\*sum1[3])-(sum1[1]\*sum1[2])))+26)%26;

if (i==0) {

det += (key[i][j]\*cofactor[i][j])%26;

}

}

det = det%26;

}

return cofactor;

}

public static int[][] adjoint(int[][] cofactor) {

int arr[][] = new int[cofactor.length][cofactor[0].length];

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

for (int j=0; j<arr[0].length; j++) {

arr[i][j] = cofactor[j][i];

}

}

return arr;

}

public static int[][] matMultiplication(int[][] arr, int[][] key) {

int arr2[][] = new int[key.length][arr[0].length];

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

for (int j=0; j <arr[0].length; j++) {

int sum = 0;

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

sum += key[i][k]\*arr[k][j];

}

arr2[i][j] = sum%26;

}

}

return arr2;

}

public static void display(int[][] arr) {

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

for (int j=0; j<arr[0].length; j++) {

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

}

System.out.println();

}

}

public static int[][] numEncoding(String word) {

int r = 3;

int c = word.length()/3;

int arr[][] = new int[r][c];

int iter = 0;

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

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

arr[j][i] = word.charAt(iter++)-97;

}

}

return arr;

}

public static String charEncoding(int[][] arr) {

String word = "";

for (int i=0;i<arr[0].length;i++) {

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

word += (char)(arr[j][i]+97);

}

}

return word;

}

}

**INTRUSION:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

#Read CSV FIle

data = pd.read\_csv("/content/kddcup99\_csv.csv")

data = pd.read\_csv("/content/kddcup99\_csv.csv", na\_values=[' ?'])

missing=data[data.isnull().any(axis=1)]

data1=data.dropna(axis=0)

data1['label']=data['label']

new\_data = pd.get\_dummies(data1,drop\_first=True)

final\_list=list(new\_data.columns)

features=list(set(final\_list)-set(['label']))

#Seperate attributes and class variables

x=new\_data[features].values

y=new\_data['label'].values

#Seperate training and testing data

train\_x, test\_x, train\_y, test\_y=train\_test\_split(x,y,test\_size=0.4)

#Train model

l=LogisticRegression(max\_iter = 20000)

l.fit(train\_x, train\_y)

#Test model

predict2=l.predict(test\_x)

#Accuracy

a=accuracy\_score(test\_y, predict2)

print("Accuracy : ",a)

**EULER TOTIANT:**

import math

p = int (input("enter : "))

l=[]

for i in range(1,p):

l.append(i)

print("residual ",l)

e=[]

for i in range(1,p):

if(math.gcd(i,p)==1):

e.append(i)

print("euler ",e)

**RSA:**

import math

#key generation

p= int(input("enter p : "))

q= int(input("enter q : "))

n=p\*q

print("n = ",n)

phi\_n = ((p-1)\*(q-1))

print("phi\_n = ",phi\_n)

#calculate e ==> gcd(e,phi\_n)=1

for i in range(2,phi\_n):

if((math.gcd(i,phi\_n)==1)):

e = i

break

else:

continue

print("e = ",i)

#calculate d ===> d\*e mod(phi\_n) =1

for i in range(1,phi\_n):

if(((((i%phi\_n)\*(e%phi\_n))%phi\_n)==1)):

d = i

break

else:

continue

print("d = ",d)

def encryption(M,e,n):

#C = ((M\*\*e)%n)

C= pow(M,e,n)

return C

def decryption(C,d,n):

#M = ((C\*\*d)%n)%26

M= pow(C,d,n)

plain=txt[M]

return plain

txt ="abcdefghijklmnopqrstuvwxyz"

choice = int(input("enter 0 - exit/n 1-encryption/n2 - decryption/n"))

while(choice!=0):

if(choice==1):

msg=input(("enter the plain text(M) : "))

M= txt.index(msg)

ciphertxt=encryption(M,e,n)

print("Cipher txt of ",msg," is ",ciphertxt)

elif(choice==2):

C=int(input("enter the cipher text(C) : "))

plaintxt=decryption(C,d,n)

print("Plain text of ",C," is ",plaintxt)

else:

print("wrong choice")

choice = int(input("enter 0 - exit/n 1-encryption/n2 - decryption/n"))

**OPEN SSL:**

Open ssl> x64>bin>create text file with name and reg num

openssl aes-128-cbc -e -in plaintxt.txt -out ciphertxt.bin -k "password" -nosalt

openssl aes-128-cbc -d -in ciphertxt.bin -out pt.txt -k "password" –nosalt

openssl genrsa –out privatekey.pem

openssl rsa -pubout -in privatekey.pem -out publickey.pem

openssl rsa -text -in privatekey.pem

openssl rsautl -encrypt -in plaintxt.txt -pubin -inkey publickey.pem -out crsa.bin

openssl rsautl -decrypt -in crsa.bin -inkey privatekey.pem -out prsa.txt

openssl md5 plain.txt

openssl SHA256 plaintxt.txt

openssl dgst -sha1 -sign privatekey.pem -out s.bin plaintxt.txt

openssl dgst -sha1 -verify publickey.pem -signature s.bin plaintxt.txt

Wireshark:

1)version of SSL is supported by the client?

CLIENT HELLO: sslv2

2) cryptographic algorithms supported by the client in Client Hello message

CLIENT HELLO: cipher suites

3) List the various parameters present in the public key certificate of the server

Server hello when certificate is done

4) Identify the public key of the server

Server hello when certificate is done

5. Identify the length of key exchanged by the Client?

Client key exchange,change cipher spec..

Length

6) What algorithm is used for encrypting the session key?

Client key exchange,change cipher spec..

Algorithm

7) List the various parameters specified in the Encrypted handshake message

Client key exchange,change cipher spec..

8. Calculate the time taken for completion of the entire handshake protocol.

Client hello -change cipher spec