import numpy as np

import string

# for the keyword

def key1():

##print("key:")

word = input()

alph="abcdefghijklmnopqrstuvwxyz"

if len(word)!=len(word2) :

for i in range((len(word2)-len(word))):

word=word+alph[i]

##print(word)

return word

else:

return word

def plain1():

##print("plaintxt:")

word2 = input()

th=len(word2)

return word2,th

def encrypt(word,word2,th):

keytxt = []

plaintxt = []

##p1 = len(word2)

rc1 = th

if np.sqrt(rc1) - int(np.sqrt(rc1)) == 0:

modHelper1 = np.sqrt(rc1)

else:

modHelper1 = int(np.sqrt(rc1) + 1)

rows1 = np.int(modHelper1)

cols1 = np.int(modHelper1)

q = (rows1 \* cols1) - rc1

pl = "kgfbahubaliffgentlemanmcaninnukorispidermanincrediblegullyboyseethammavakitlosirimallechettuvvrrangammaekkadikipothavchinnavaadabhalemogaduvoyrajugarigadhijambalakadipambasahoorowdybabymaarimaswhythiskolaveridimangammaramadasdevadasannavaanazebrahrudhayamloveillalupriyudunpremishqkadhalviswaroopamdasavatharammamidiakuluuyyalajampalabhagumathichennaiexpressagnipathsaahasamswasagasagipoosamsaramokachadarangamkathanayakudumahanatidangaldongamoguduneneraajumanthrikrishnamvandhejagadgurumvasthavayyakhaidhishivaremobirlaparugu"

for i in range(q):

word2 = word2 + pl[i]

alph = "qwertyuiopasdfghjklzxcvbnmqawsedrftgyhujikolzxcvbnmpasdewqsfrvbhunjklmnbvgfdcxsxcvbhnzsdcxvfgbnhjmklramcharanupasanaalluarjunsnehamaheshnamrathantrpranathiariyanaviyaarhaayushadvaithvihaanakshajabhiramnagarjunaamalanagachaithanyasamanthaakhilsnehaprasanadhanushsowndaryananianjanachigurakulochilukammadeepammayabazarrajinikanthravitejarajamoulipurijagannathnandinimanasulounnadhicheppalaniunnadhimataluraveilaundipooradhegundeyneeveleylamborghinioosupodhuurukodvachindhammadevarkondatelusapreminchananidhamdhanandamneelacherindhinanneveyyelanubandm"

if len(word) != len(word2):

for i in range((len(word2) - len(word))):

word = word + alph[i]

##print(word)

##return word

##else:

##return word

#

for character in word:

number = ord(character) - 96

keytxt.append(number)

##print("keytext is \n", keytxt)

for character in word2:

number = ord(character) - 96

plaintxt.append(number)

##print("plaintxt is \n",plaintxt)

rc=len(word)

rc1=len(word2)

##print("length of key \n",rc)

##print("lenghth of plaintxt \n ",rc1)

modHelper = np.sqrt(rc)

rows, cols = np.int(modHelper), np.int(modHelper)

##print("rows and cols of key \n",rows,cols)

modHelper1 = np.sqrt(rc1)

rows1, cols1 = np.int(modHelper), np.int(modHelper)

##print("rows and cols of plaintxt \n",rows1,cols1)

new = np.reshape(keytxt, (rows, cols))

##print("key matrix is \n",new)

new1=np.reshape(plaintxt,(rows1,cols1))

##print("plaintext matrix is \n",new1)

result = [[0 for row in range(len(new1))] for col in range(len(new[0]))]

for i in range(len(new1)):

# iterating by coloum by B

for j in range(len(new[0])):

# iterating by rows of B

for k in range(len(new)):

result[i][j] += new1[i][k] \* new[k][j]

#for r in result:

##print("result \n",result)

new2=np.reshape(result,(rows1,cols1))

##print("result is \n",new2)

cofactor=np.linalg.inv(new2).T \* np.linalg.det(new2)

#cofactor=cofacto.astype(int)

##print("cofactor matrix is \n",cofactor)

CT = [[0 for row in range(len(cofactor))] for col in range(len(cofactor[0]))]

for i in range(len(cofactor)):

#iterate through columns

for j in range(len(cofactor[0])):

CT[j][i] = cofactor[i][j]

#for r in CT:

# print("transpose is \n",r)

ne=np.reshape(CT,(cols1,rows1))

##print("transpose is \n",ne)

cm1=ne%26

cm=cm1.astype(int)

##print("cipher matrix is\n",cm)

ciphertext = []

for i in range(rows):

for j in range(len(cm[0])):

ciphertext.append(chr(cm[i][j] + 97))

##return ''.join(ciphertext),ne,np.linalg.det(result)

##print("ciphertext ","".join(ciphertext))

##print("encryption is done")

for i in range(th):

return ''.join(ciphertext[i:i+th]),ne,np.linalg.det(result)

def decrypt(word,ne,a,th):

##print(th)

##modHelper1 = np.sqrt(rc1)

##rows1, cols1 = np.int(modHelper1), np.int(modHelper1)

keytxt = []

alph = "qwertyuiopasdfghjklzxcvbnmqawsedrftgyhujikolzxcvbnmpasdewqsfrvbhunjklmnbvgfdcxsxcvbhnzsdcxvfgbnhjmklramcharanupasanaalluarjunsnehamaheshnamrathantrpranathiariyanaviyaarhaayushadvaithvihaanakshajabhiramnagarjunaamalanagachaithanyasamanthaakhilsnehaprasanadhanushsowndaryananianjanachigurakulochilukammadeepammayabazarrajinikanthravitejarajamoulipurijagannathnandinimanasulounnadhicheppalaniunnadhimataluraveilaundipooradhegundeyneeveleylamborghinioosupodhuurukodvachindhammadevarkondatelusapreminchananidhamdhanandamneelacherindhinanneveyyelanubandm"

if len(word) != (len(ne)\*len(ne[0])):

for i in range((len(ne)\*len(ne[0])-len(word))):

word = word + alph[i]

##print(word)

##return word

##else:

##return word

rc = len(word)

##rc1 = len(word2)

modHelper = np.sqrt(rc)

rows, cols = np.int(modHelper), np.int(modHelper)

for character in word:

number = ord(character) - 96

keytxt.append(number)

new = np.reshape(keytxt, (rows, cols))

## print("new",new)

##new1 = np.reshape(plaintxt, (rows1, cols1))

cofactor1 = np.linalg.inv(ne).T \* np.linalg.det(ne)

#cofactor1=cofactor1.astype(int)

##print("cofactor matrix is \n",cofactor1)

CT1 = [[0 for row in range(len(cofactor1))] for col in range(len(cofactor1[0]))]

for i in range(len(cofactor1)):

#iterate through columns

for j in range(len(cofactor1[0])):

CT1[j][i] = cofactor1[i][j]

#for r in CT1:

# print("transpose is \n",r)

ne1=np.reshape(CT1,(cols,rows))

##print("transpose is \n",ne1)

##a=np.linalg.det(result)

b=a\*\*(rows-2)

##print("det",b)

ne1=ne1/b

##print("adj",ne1)

ne3=np.linalg.inv(new)

# .inv(new)

##print("inverse is",ne3)

result1 = [[0 for row in range(len(ne1))] for col in range(len(ne3[0]))]

for i in range(len(ne1)):

# iterating by coloum by B

for j in range(len(ne3[0])):

# iterating by rows of B

for k in range(len(ne3)):

result1[i][j] += (ne1[i][k])\*(ne3[k][j])

#for r in result:

##print("result \n",result1)

new3=np.reshape(result1,(rows,cols))

##print("result is \n",new3)

new4=np.around(new3)

##print("res is",new4)

new5=new4.astype(int)

##print("res1",new5)

plaintext = []

for i in range(rows):

for j in range(len(new5[0])):

plaintext.append(chr(new5[i][j] + 96))

##return ''.joinQ2A(plaintext)

# print("plaintext ","".join(plaintext))

##print("decryption is done")

for i in range(th):

#print("plaintext ",plaintext[i:i+th])

return ''.join(plaintext[i:i+th])

def GUI():

try:

import Tkinter as tk

except:

import tkinter as tk

window = tk.Tk()

window.title('Data Security')

window.geometry("800x150")

def encodeandshow():

e,t,d=encrypt(e1.get(),e2.get(),len(e2.get()))

e3.insert(0,e)

return t,d

def decodeandshow():

e,t,d=encrypt(e1.get(),e2.get(),len(e2.get()))

f=decrypt(e1.get(),t,d,len(e2.get()))

e4.insert(0,f)

l1 = tk.Label(window, text="Enter key")

l1.grid(row=0, column=0)

titletext1 = tk.StringVar()

e1 = tk.Entry(window, textvariable=titletext1, width=100, bg='LightBlue')

e1.grid(row=0, column=1)

l2 = tk.Label(window, text="Enter text")

l2.grid(row=1, column=0)

titletext2 = tk.StringVar()

e2 = tk.Entry(window, textvariable=titletext2, width=100, bg='LightBlue')

e2.grid(row=1, column=1)

l3 = tk.Label(window, text="Cipher text")

l3.grid(row=2, column=0)

titletext3 = tk.StringVar()

e3 = tk.Entry(window, textvariable=titletext3, width=100, bg='LightBlue')

e3.grid(row=2, column=1)

e3.bind('<e>', encodeandshow)

b1 = tk.Button(window, text="Encrypt", bg='yellow', command=encodeandshow)

b1.grid(row=4, column=0)

l4 = tk.Label(window, text="Plain text")

l4.grid(row=3, column=0)

titletext4 = tk.StringVar()

e4 = tk.Entry(window, textvariable=titletext4, width=100, bg='LightBlue')

e4.grid(row=3, column=1)

e4.bind('<e>', decodeandshow)

b2 = tk.Button(window, text="Decrypt", bg='yellow', command=decodeandshow)

b2.grid(row=4, column=1)

def Reset():

e1.delete(0,'end')

e2.delete(0,'end')

e3.delete(0,'end')

e4.delete(0,'end')

b3 = tk.Button(window, text="reset", bg='yellow', command=Reset)

b3.grid(row=4, column=2)

tk.mainloop()

def win1():

try:

import Tkinter as tk

except:

import tkinter as tk

window = tk.Tk()

window.configure(bg='black')

window.title('Data Security')

window.geometry("1000x600")

l0 = tk.Label(window, text=" WELCOME TO FUSION-PROJECT EXPO",font='Algerian 26 bold',height=2,bg='black',foreground='magenta')

l0.grid(row=1, column=1)

l1 = tk.Label(window, text=" DATA SECURITY",font='Jokerman 26 bold',bg='black',foreground='green')

l1.grid(row=2, column=1)

l1 = tk.Label(window, text=" ", height=4, bg='black')

l1.grid(row=3, column=1)

l1 = tk.Label(window, text=" Team members :", font=' Stencil 20 bold',height=2, bg='black',foreground='white')

l1.grid(row=4, column=1)

l1 = tk.Label(window, text=" K.Vyshnavi",font='Forte 18 ', bg='black',foreground='white')

l1.grid(row=5, column=1)

l1 = tk.Label(window, text=" B.Dikxitha",font='Forte 18 ', bg='black',foreground='white')

l1.grid(row=6, column=1)

l1 = tk.Label(window, text=" K.Vinathi", font='Forte 18 ', bg='black',foreground='white')

l1.grid(row=7, column=1)

l1 = tk.Label(window, text=" ",height=4, bg='black')

l1.grid(row=8, column=1)

def new\_window():

GUI()

#window.destroy()

b0 = tk.Button(window, text="next>>", bg='white', command=new\_window)

b0.grid(row=9, column=2)

tk.mainloop()

win1()