#GUI Imports

from asyncio.windows\_events import NULL

from msilib.schema import RadioButton

import tkinter as tk

from tkinter import Radiobutton, Variable, messagebox, filedialog, font, ttk

import os

import random

# DES Imports

from Crypto.Cipher import DES

# AES Imports

from Crypto.Cipher import AES

#hash

import hashlib

#Global

file\_flag = False

file\_path = ''

file\_name = ''

key = b''

iv = b''

size = 2048

#main

def run\_code():

if var1.get() == "Default":

if var2.get() == "Encrypt":

update\_progress(1)

file\_path = entry0.get()

update\_progress(3)

file\_name = os.path.basename(file\_path)

update\_progress(6)

xoe\_Encrypt(file\_name,text\_to\_int(entry1.get()))

update\_progress(10)

listbox2.items.add("File encrypted successfully")

if var2.get() == "Decrypt":

update\_progress(1)

file\_path = entry0.get()

update\_progress(3)

file\_name = os.path.basename(file\_path)

update\_progress(6)

xor\_Decrypt(file\_name, text\_to\_int(entry1.get()))

update\_progress(10)

listbox2.items.add("File decrypted successfully")

if var1.get() == "Custom":

if var2.get() == "Encrypt":

update\_progress(1)

#print(entry1.get())

#print(rot47\_encode(entry1.get()))

#print(hash\_keyword16(rot47\_encode(entry1.get())))

file\_path = entry0.get()

update\_progress(3)

#print(file\_path)

AES\_encrypt(file\_path,hash\_keyword16(rot47\_encode(entry1.get())))

update\_progress(5)

convert\_file\_16\_8()

update\_progress(7)

file\_name = os.path.basename(file\_path)

update\_progress(9)

DES\_encrypt(hash\_keyword8(rot47\_encode(entry1.get())),file\_name)

update\_progress(10)

#Remove\_file(file\_path)

listbox1.insert(tk.END,"File encrypted successfully")

if var2.get() == "Decrypt":

update\_progress(1)

file\_path = entry0.get()

update\_progress(3)

file\_name = os.path.basename(file\_path)

update\_progress(5)

AES\_decrypt(file\_name,hash\_keyword16(rot47\_encode(entry1.get())),iv)

update\_progress(7)

convert\_file\_8\_16()

update\_progress(9)

DES\_decrypt(hash\_keyword8(rot47\_encode(entry1.get())),file\_name)

update\_progress(10)

listbox2.insert(tk.END,"File decrypted successfully")

#xor

def xoe\_Encrypt(filename, key):

file = open(filename, "rb")

data = file.read()

file.close()

data = bytearray(data)

for index, value in enumerate(data):

data[index] = value ^ key

file = open("CC-" + filename, "wb")

file.write(data)

file.close()

def xor\_Decrypt(filename, key):

file = open(filename, "rb")

data = file.read()

file.close()

data = bytearray(data)

for index, value in enumerate(data):

data[index] = value ^ key

file = open(filename, "wb")

file.write(data)

file.close()

#AES encrypt & decrypt #Skyf

def AES\_encrypt(file\_path,key1):

iv = ''.join([chr(random.randint(0, 0xFF)) for i in range(16)])

key = b'Sixteen byte key'

cipher = AES.new(key, AES.MODE\_EAX)

#print(get\_file\_bytes)

nonce = cipher.nonce

ciphertext, tag = cipher.encrypt\_and\_digest(get\_file\_bytes(file\_path))

with open('encfase1.bin', 'wb') as fout:

fout.write(nonce)

fout.write(ciphertext)

#print(ciphertext)

def AES\_decrypt(file\_path, key1, nonce):

with open('encfase1.bin', 'rb') as fin:

nonce = fin.read(16)

ciphertext = fin.read()

cipher = AES.new(key1, AES.MODE\_EAX, nonce)

plaintext = cipher.decrypt(ciphertext)

with open('DD-'+file\_path, 'wb') as fout:

fout.write(plaintext)

def DES\_encrypt(key3,file\_name):

cipher = DES.new(key3, DES.MODE\_OFB)

with open('encfase2.bin', 'rb') as input\_file, open('CC-' + file\_name, 'wb') as output\_file:

output\_file.write(cipher.iv + cipher.encrypt(input\_file.read()))

return 'CC-' + file\_name

def DES\_decrypt(key3,file\_name):

cipher = DES.new(key3, DES.MODE\_OFB)

# Open the input and output files

with open(file\_name, 'rb') as input\_file, open('decfase1.bin', 'wb') as output\_file:

# Read the input file in blocks of 8 bytes

while True:

block = input\_file.read(8)

if not block:

break # Reached end of file

# Decrypt the block and write it to the output file

decrypted\_block = cipher.decrypt(block)

output\_file.write(decrypted\_block)

def text\_to\_int(text\_input):

byte\_str = text\_input.encode('ascii')

byte\_int = int.from\_bytes(byte\_str, 'big')

return byte\_int % 256 + 1

def convert\_file\_8\_16():

with open('decfase1.bin', 'rb') as input\_file, open('decfase2.bin', 'wb') as output\_file:

# Read the input file in blocks of 8 bytes

while True:

block = input\_file.read(8)

if not block:

break # Reached end of file

# Repeat the block to create a 16-byte block

block = block + block

# Write the 16-byte block to the output file

output\_file.write(block)

# 8 byte to 16 byte help my asseblief

def convert\_file\_16\_8():

with open('encfase1.bin', 'rb') as input\_file, open('encfase2.bin', 'wb') as output\_file:

# Read the input file in blocks of 16 bytes

while True:

block = input\_file.read(16)

if not block:

break # Reached end of file

# Split the block into two 8-byte blocks

block1 = block[:8]

block2 = block[8:]

# Write the two blocks to the output file

output\_file.write(block1)

output\_file.write(block2)

# ROT47 pass Encrypt #Skyf

def rot47\_encode(keyword):

encoded\_keyword = ""

for char in keyword:

ascii\_code = ord(char)

if 33 <= ascii\_code <= 126:

encoded\_ascii\_code = (ascii\_code - 33 + 47) % 94 + 33

encoded\_keyword += chr(encoded\_ascii\_code)

else:

encoded\_keyword += char

return encoded\_keyword

def decode\_rot47(keyword):

# """Decode a keyword using ROT47 encryption"""

# Initialize an empty result string

result = ""

# Loop through each character in the keyword

for c in keyword:

# Get the ASCII code of the character

ascii\_code = ord(c)

# If the character is in the range of printable ASCII characters

if 33 <= ascii\_code <= 126:

# Decode the character using the ROT47 algorithm

decoded\_char = chr(33 + ((ascii\_code + 14) % 94))

else:

# Otherwise, the character is not in the range of printable ASCII characters,

# so just append it as is to the result string

decoded\_char = c

# Append the decoded character to the result string

result += decoded\_char

# Return the decoded result

return result

#hash 16

def hash\_keyword16(keyword):

# Create a SHA-256 hash object

sha256 = hashlib.sha256()

# Update the hash object with the keyword

sha256.update(keyword.encode())

# Get the first 13 bytes of the hash as bytes

hash\_bytes = sha256.digest()[:16]

return hash\_bytes

#hash 8

def hash\_keyword8(keyword):

# Create a SHA-256 hash object

sha256 = hashlib.sha256()

# Update the hash object with the keyword

sha256.update(keyword.encode())

# Get the first 8 bytes of the hash as bytes

hash\_bytes = sha256.digest()[:8]

return hash\_bytes

# locked file 1 pass 2 file #Jaap

#source file delete #Skyf

def Remove\_file(file\_path):

if os.path.exists(file\_path):

os.remove(file\_path)

listbox1.insert(tk.END,"File removed successfully")

else:

listbox1.insert(tk.END,"File could not be removed")

# integration #Skyf en Jaap

# get file bytes

def get\_file\_bytes(file\_path):

with open(file\_path, 'rb') as input\_file:

file\_bytes = input\_file.read()

return file\_bytes

#gui #Jaap

def get\_file():

#file\_flag = True

file\_path = filedialog.askopenfilename()

file\_type = os.path.splitext(file\_path)[1]

entry0.delete(0, tk.END)

entry0.insert(0, file\_path)

root = tk.Tk(className='Encryption Project')

root.geometry("500x450")

root.configure(background='#000000')

frame = tk.Frame(master=root, background='#000000', relief=tk.GROOVE, bd=1)

frame.pack(pady=20, padx=60)

# Label styles

label\_style = {'background': '#000000', 'foreground': '#20C20E'}

groove\_style = {'relief': tk.GROOVE, 'bd': 1}

# row 0

label2 = tk.Label(master=frame, text="Shhhhhh! It's a secret", font=("", 20), \*\*label\_style)

label2.grid(row=0, column=0, columnspan=3, pady=12, padx=10)

# row 1

tk.Label(master=frame, text="File:", \*\*groove\_style, \*\*label\_style).grid(row=1, column=0, padx=10, sticky="W")

entry0 = tk.Entry(master=frame, foreground='#20C20E', background='#000000', \*\*groove\_style)

entry0.grid(row=1, column=1, padx=10, columnspan=2, sticky="W")

tk.Button(master=frame, text="Browse", bg='black', fg='#20C20E', \*\*groove\_style, command=get\_file).grid(row=1, column=2)

# row 2

tk.Label(frame, text="Choose an option:", \*\*groove\_style, \*\*label\_style).grid(row=2, column=0, padx=10, sticky="W")

options2 = ["Default", "Custom"]

var1 = tk.StringVar(value=options2[0])

for i, option in enumerate(options2):

rad1= Radiobutton(

frame, text=option, value=option, bg='#000000', fg='#20C20E',

highlightbackground='black', highlightcolor='black', selectcolor='black' ,variable=var1

).grid(row=2, column=i+1, padx=10, sticky="W")

# row 3

tk.Label(frame, text="Key:", \*\*groove\_style, \*\*label\_style).grid(row=3, column=0, padx=10, sticky="W")

entry1 = tk.Entry(master=frame, foreground='#20C20E', background='#000000', \*\*groove\_style)

entry1.insert(0, "Hello, World!")

entry1.grid(row=3, column=1, padx=10, columnspan=2, sticky="W")

# row 4

tk.Label(frame, text="Mode:", \*\*groove\_style, \*\*label\_style).grid(row=4, padx=10, sticky="W")

options3 = ["Encrypt", "Decrypt"]

var2 = tk.StringVar(value=options3[0])

for i, option in enumerate(options3):

rad2 = Radiobutton(

frame, text=option, value=option, bg='#000000', fg='#20C20E',

highlightbackground='black', highlightcolor='black', selectcolor='black', variable=var2

).grid(row=4, column=i+1, padx=10, sticky="W")

#row 5

listbox1 = tk.Listbox(master=frame, fg='#20C20E', bg='#000000',relief=tk.GROOVE, bd=1)

listbox1.grid(row=5, column=0,padx = 10,columnspan=2,sticky="W" )

tk.Button(master=frame, text="Execute", bg='black', fg='#20C20E', \*\*groove\_style, command=run\_code).grid(row=5, column=1)

listbox2 = tk.Listbox(master=frame, fg='#20C20E', bg='#000000',relief=tk.GROOVE, bd=1)

listbox2.grid(row=5, column=2 ,padx = 10,sticky="E")

#row6

def update\_progress(progress):

progress\_width = progress \* frame.winfo\_width() // 10

progressbar.coords("progress", 0, 0, progress\_width, 20)

def set\_progress(num):

if num < 1:

num = 1

elif num > 10:

num = 10

progress\_percent = (num - 1) \* 10 # calculate progress percentage

update\_progress(progress\_percent / 100) # update progress bar

progressbar = tk.Canvas(frame,height=20, bg='#000000', highlightthickness=0, relief=tk.GROOVE, bd=1)

progressbar.grid(row=6, column=0,columnspan=3 , sticky="W")

progressbar.create\_rectangle(0, 0, 0, 20, fill='#20C20E', width=0, tags="progress")

# Cal

root.mainloop()