**Un conjunto de letras negras en un fondo negro

Descripción generada automáticamente con confianza bajaIcono

Descripción generada automáticamente con confianza media**

**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Ingeniería en Sistemas Computacionales**

**Rodríguez Olmos Noé**

**Hernández López Bryan**

**Dra. Nidia A. Cortez Duarte**

**Práctica 1: AE Y AEE.**

**Grupo: 6CM1**

**Criptografía|Introduction To Cryptography**

**24 de marzo del 2023**

Resumen

En esta práctica se realizó un programa basado en el Algoritmo de Euclides, sirve para el cifrador Affine que se estará programando próximamente, gracias a lo visto en clase, verificamos que el programa con el Algoritmo de Euclides funcione realizando las pruebas con diferentes números de distintas longitudes. El programa realizado tiene una interfaz grafica con la cual permite al usuario introducir los tres parámetros solicitados, también el programa indica con un mensaje cuando los números no son coprimos.

El programa valida los valores introducidos y arroja la función de cifrado, así como las dos funciones de descifrado.

Algoritmo de Euclides y Algoritmo de Euclides Extendido (AE y AEE).

El algoritmo de Euclides se utiliza para calcular el máximo común divisor (MCD o gcd) de dos números enteros positivos. El MCD (gcd) es el número más grande que divide exactamente a ambos números y puede ser útil en una variedad de situaciones, como en la simplificación de fracciones o en la resolución de ecuaciones lineales.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamentePor otro lado, el algoritmo de Euclides extendido es una versión ampliada del algoritmo de Euclides que permite además expresar al máximo común divisor como una combinación lineal, mediante las combinaciones lineales, al despejar los residuos nos permite obtener el inverso multiplicativo, y al mismo tiempo conocer el MCD (gcd).

Fig. 1 Algoritmo de Euclides

Código

# ---------------- LIBRERIAS ---------------- #

from tkinter import \*

import tkinter as tk

from tkinter import ttk

import os

import numpy as np

import math

# ------------ GENERACION DE LA VENTANA ------------ #

ventana = tk.Tk()

ventana.geometry("600x450")

ventana.title("Practica #1 - Cifrado Afin")

# ------------ VARIABLES GLOBALES ------------ #

valorN = tk.IntVar()

valorA = tk.IntVar()

valorB = tk.IntVar()

# ------------ CONFIGURACION GRID VENTANA ------------ #

ventana.rowconfigure(0, weight=1)

ventana.rowconfigure(1, weight=1)

ventana.rowconfigure(2, weight=1)

ventana.rowconfigure(3, weight=1)

ventana.rowconfigure(4, weight=1)

ventana.rowconfigure(5, weight=1)

ventana.rowconfigure(6, weight=1)

ventana.rowconfigure(7, weight=1)

ventana.rowconfigure(8, weight=1)

ventana.columnconfigure(0, weight=1)

ventana.columnconfigure(1, weight=1)

ventana.columnconfigure(2, weight=1)

ventana.columnconfigure(3, weight=1)

ventana.columnconfigure(4, weight=1)

ventana.columnconfigure(5, weight=1)

# ------------ ALGORITMO EXTENDIDO DE EUCLIDES ------------ #

def algoritmoExtendidoEuclides(a, b):

    if (a == 0):

        return 0, 1

    else:

        x, y = algoritmoExtendidoEuclides(b % a, a)

        return y-(b//a)\*x, x

# ------------ ALGORITMO DE EUCLIDES ------------ #

def algoritmoEuclides():

    textoCarita = tk.Label(ventana, text="")

    textoCarita.grid(row=3, column=0, sticky="NSWE", columnspan=6)

    if (valorN.get() == 0 and valorA.get() == 0):

        textoCarita.config(text="ESPERANDO CAPTURA DE VALORES", fg="grey", font="Helvetica 14 italic")

        textoFuncionCifrado = tk.Label(ventana, text="")

        funcionCifrado = tk.Label(ventana, text="")

        textoFuncionDesifrado = tk.Label(ventana, text="")

        funcionDesifrado1 = tk.Label(ventana, text="")

        funcionDesifrado2 = tk.Label(ventana, text="")

        textoFuncionCifrado.grid(row=4, column=0, sticky="NSWE", columnspan=6)

        funcionCifrado.grid(row=5, column=0, sticky="NSWE", columnspan=6)

        textoFuncionDesifrado.grid(row=6, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado1.grid(row=7, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado2.grid(row=8, column=0, sticky="NSWE", columnspan=6)

    elif (math.gcd(valorN.get(), valorA.get()) == 1):

        textoCarita.config(text="VALOR EXISTENTE  ヽ(•‿•)ノ", fg="green", font="Helvetica 14 italic")

        x, y = algoritmoExtendidoEuclides(valorA.get(), valorN.get())

        print(f'x = {x}, y = {y}')

        textoFuncionCifrado = tk.Label(ventana, text="- Funcion de Cifrado -", fg="black", font="Helvetica 12 bold")

        cadenaCifrado = "C = " + str(valorA.get()) + "p + " + str(valorB.get()) + " mod " + str(valorN.get())

        funcionCifrado = tk.Label(ventana, text=cadenaCifrado, fg="black", font="Helvetica 10")

        textoFuncionDesifrado = tk.Label(ventana, text="- Funcion de Desifrado -", fg="black", font="Helvetica 12 bold")

        cadenaDescifrado1 = "P = " + str(x) + " (C + " + str(valorN.get()-valorB.get()) + ") mod " + str(valorN.get())

        funcionDesifrado1 = tk.Label(ventana, text=cadenaDescifrado1, fg="black", font="Helvetica 10")

        cadenaDescifrado2 = "P = " + str(x) + "C + " + str((x\*(valorN.get()-valorB.get()))%valorN.get()) + " mod " + str(valorN.get())

        funcionDesifrado2 = tk.Label(ventana, text=cadenaDescifrado2, fg="black", font="Helvetica 10")

        textoFuncionCifrado.grid(row=4, column=0, sticky="NSWE", columnspan=6)

        textoFuncionDesifrado.grid(row=6, column=0, sticky="NSWE", columnspan=6)

        funcionCifrado.grid(row=5, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado1.grid(row=7, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado2.grid(row=8, column=0, sticky="NSWE", columnspan=6)

    else:

        textoCarita.config(text="VALOR NO EXISTENTE ( ╥\_╥ ) , INTENTE CON OTRO", fg="red", font="Helvetica 14 italic")

        textoFuncionCifrado = tk.Label(ventana, text="")

        funcionCifrado = tk.Label(ventana, text="")

        textoFuncionDesifrado = tk.Label(ventana, text="")

        funcionDesifrado1 = tk.Label(ventana, text="")

        funcionDesifrado2 = tk.Label(ventana, text="")

        textoFuncionCifrado.grid(row=4, column=0, sticky="NSWE", columnspan=6)

        funcionCifrado.grid(row=5, column=0, sticky="NSWE", columnspan=6)

        textoFuncionDesifrado.grid(row=6, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado1.grid(row=7, column=0, sticky="NSWE", columnspan=6)

        funcionDesifrado2.grid(row=8, column=0, sticky="NSWE", columnspan=6)

# ------------ FUNCION PRINCIPAL ------------ #

def main():

    # Texto para los campos

    textoCampos = tk.Label(ventana, text="Capture los valores de los parametros n, alpha y beta:", fg="blue", font="Helvetica 15 bold")

    textoCampos.grid(row=0, column=0, sticky="NSWE", columnspan=6)

    # Campo para el valor de N

    textoN = tk.Label(ventana, text="n : ", fg="black", font="Helvetica 12 bold")

    textoN.grid(row=1, column=0, sticky="NSWE")

    entradaN = tk.Entry(ventana, textvariable=valorN,

                        width=20, justify=tk.CENTER, state=NORMAL)

    entradaN.grid(row=1, column=1)

    # Campo para el valor de A (alpha)

    textoA = tk.Label(ventana, text="α : ", fg="black", font="Helvetica 12 bold")

    textoA.grid(row=1, column=2, sticky="NSWE")

    entradaA = tk.Entry(ventana, textvariable=valorA,

                        width=20, justify=tk.CENTER, state=NORMAL)

    entradaA.grid(row=1, column=3)

    # Campo para el valor de B (beta)

    textoB = tk.Label(ventana, text="β : ", fg="black", font="Helvetica 12 bold")

    textoB.grid(row=1, column=4, sticky="NSWE")

    entradaB = tk.Entry(ventana, textvariable=valorB,

                        width=20, justify=tk.CENTER, state=NORMAL)

    entradaB.grid(row=1, column=5)

    # Boton para verificar por el AE

    botonVerificar = tk.Button(ventana, text="V E R I F I C A R", fg="white", bg="black", font="Helvetica 12 bold", command=algoritmoEuclides)

    botonVerificar.grid(row=2, column=2, sticky="NSWE",padx=10, pady=10, columnspan=2)

    # Ejecucion en bucle de la interfaz

    ventana.mainloop()

# ------------ INICIO DEL PROGRAMA ------------ #

main()

Pruebas

Valores de prueba

* Interfaz de usuario gráfica, Texto, Aplicación

  Descripción generada automáticamente>α = 5 β=61 n = 30 y después α = 17

Fig. 2 y Fig. 3 Primera prueba de la practica 1

* α = 117 β=1 n = 244

Fig. 4 Segunda prueba de la practica 1

* Interfaz de usuario gráfica, Texto, Aplicación

  Descripción generada automáticamenteα = 97 β=12 n = 239

Fig. 5 Tercera prueba de la practica 1

* α = 11111 β=500 n = 12345

Fig. 6 Cuarta prueba de la practica 1

* α = 13 β=555 n = 99991

Fig. 7 Quinta prueba de la practica 1

* α = 10009 β=99999 n = 104729

Fig. 7 Sexta prueba de la practica 1

Conclusión.

La realización de esta práctica no fue complicado, el único detalle que se nos presentó fue la personalización de la interfaz de la aplicación y que cumpliera con todos los requisitos, debido a que nuestros conocimientos para desarrollar interfaces eran nulos o no muy buenos, así que para poder hacer la tuvimos que ver videos de YouTube introductorios a interfaces en Python. Esta practica nos va a ser de utilidad para crear el cifrador Affine, en la próxima practica y ayudar a disminuir el nivel de complejidad.

Texto

Descripción generada automáticamente con confianza baja

1

1

5

1

1

1

5

Bibliografía

* Cruise, B. (2022) Khan Academy: Computer science: Cryptorgraphy. https://es.khanacademy.org/computing/computer-science
* Nakov S. (2018). Practical Cryptography for Developers- Free Book. https://wizardforcel.gitbooks.io/practical-cryptography-for-developersbook/content/
* Kaufman, C, Perlman R., Speciner, M., Perlner, R. 2022 Network Security. Private communications in a public world Addison-Wesley Proffesional/978 -0136643609
* Menezes, A. y Vanstone, S. 1996 Handbook of Applied Cryptography CRC Press/ 978084938523