INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INTRODUCTION TO CRYPTOGRAPHY

ING. EN SISTEMAS COMPUTACIONALES

Práctica 02:

"Algoritmo Euclides y Algoritmo Extendido de Euclides”

Aplicar el funcionamiento del Algoritmo de Euclides y el Algoritmo Extendido de Euclides para encontrar las funciones de cifrado (E_k) y descifrado (D_k).

Franco Olvera Demian Oder – 2021630278

Ramos Magaña Miguel Ángel – 2021630624

Grupo: 6CM2

Profesora: Nidia Asunción Cortez Duarte

Fecha de entrega: 13/10/2023

Marco teórico:

El Algoritmo de Euclides es un algoritmo matemático que se utiliza para calcular el Máximo Común Divisor de dos números a & b a través de la descomposición de uno a partir del otro: a = bq + r, donde q equivale a a/b (a dividido por b) y r es el residuo de la operación previamente mencionada.
Para continuar el algoritmo, se aplican las ecuaciones a = b & b = r, y se reinicia el desarrollo. El algoritmo se da por concluido cuando r equivale a 0, y el valor previo que tuvo r antes de terminar es considerado el MCD de ambos números a & b.
Por otro lado, el Algoritmo Extendido de Euclides dice que, de existir MCD(a, b), existe también una ecuación que cumple que ax + by = MCD(a, b).
El Algoritmo Extendido trabaja mediante la inversión de los pasos del AE común, mediante el cual se aplican propiedades aritméticas para encontrar una combinación lineal que cumpla con ax + by = MCD(a, b). La ecuación previa recibe el nombre de Identidad de Bézout, y tanto a x como a y se le conocen como Coeficientes de Bézout.
Trabajando regresivamente, se han encontrado ecuaciones que calculan los coeficientes que satisfacen dicha ecuación, las cuales son:

- $x_i = x_{i-2} - q_i(x_{i-1})$
- $x_i = x_{i-2} - q_i(x_{i-1})$

Donde q es el cociente que resulta de hacer a/b repetidamente, al igual que el Algoritmo de Euclides convencional. De esta manera, es posible desarrollar el algoritmo normalmente mientras se van calculando los Coeficientes de Bézout. Al tener que acceder a datos previos a la implementación, se asume que:

- $x_{i-2} = 1$; $x_{i-1} = 0$
- $y_{i-2} = 0$; $y_{i-1} = 1$

Al aplicar las ecuaciones conjuntamente con el Algoritmo de Euclides, se calculan los coeficientes al final simultáneamente.

Objetivo:

Implementar en equipo funciones que permitan validad si un valor de Alpha tiene inverso multiplicativo (Algoritmo de Euclides) y encontrar dicho valor (Algoritmo Extendido de Euclides).

Instrucciones:

- Crear una interfaz gráfica en el lenguaje de programación de su preferencia que permita solicitar al usuario los valores de $\boldsymbol{n}, \boldsymbol{\alpha}$ *y* $\boldsymbol{\beta}$.
- Verifica que el valor de $\boldsymbol{\beta}$ pertenezca al conjunto (0, n].
- Implementar el algoritmo de Euclides para validar $\boldsymbol{\alpha}$, en caso de que no sea válido solicitar al usuario que introduzca un valor diferente.
- En caso de que MCD($\boldsymbol{\alpha}, \boldsymbol{n}$) = 1, calcular el inverso multiplicativo de $\boldsymbol{\alpha}$ implementando el algoritmo extendido de Euclides.

- Mostrar las funciones de cifrado (Ek) con los valores introducidos y de descifrado (Dk) con los valores calculados.

Desarrollo:

Implementación del Algoritmo de Euclides en Python:

```
# Aplicamos el Algoritmo de Euclides para detectar si los números son o no coprimos
    i = 0
    modules.append(n)
    modules.append(alpha)
    print(f"\nRealizamos: gdc({modules[i + 1]}, {modules[i]})\n")
    while 0 not in modules:
        naturalDiv.append(modules[i] // modules[i + 1])
        modules.append(modules[i] % modules[i + 1])
        equations1.append(f"{modules[i]} = {modules[i + 1]}({naturalDiv[i]}) + {modules[i + 2]}")
        equations2.append(f"{modules[i + 2]} = {modules[i]} - {modules[i + 1]}({naturalDiv[i]})")
        i += 1

    # Imprimimos las ecuaciones desarrolladas a lo largo del algoritmo
    for ecuacion in equations1:
        print(ecuacion)

    print(f"Finalmente:  gdc({modules[0]}, {modules[1]}) = {modules[len(modules) - 2]}\n")
```

Este código parece implementar el Algoritmo de Euclides para encontrar el máximo común divisor (MCD) de dos números enteros, **n** y **alpha (*α*)**, y durante el proceso, también se construyen ecuaciones que representan cada paso del algoritmo.

Implementación del Algoritmo de Euclides Extendido en Python:

```
# Preparamos las ecuaciones para el algoritmo extendido de Euclides
        print("Preparamos las ecuaciones para aplicar el algoritmo extendido: \n")
        for ecuacion in equations2[:-1]:
            print(ecuacion)

        # Realizamos el algoritmo extendido de Euclides
        alpha_copy, n_copy = alpha, n
        x = [1, 0]
        y = [0, 1]

        while n_copy != 0:
            quotient = alpha_copy // n_copy
            alpha_copy, n_copy = n_copy, alpha_copy % n_copy
            x[0], x[1] = x[1], x[0] - quotient * x[1]
            y[0], y[1] = y[1], y[0] - quotient * y[1]

        # Aislamos los valores de los coeficientes x & y
        x_final, y_final = x[0], y[0]

        # Imprimir los coeficientes x y y en la ecuación ax + by = 1
```

```
        print(f"\nLos coeficientes finales de 'x' y 'y' son: \nx = {x_final}\ny = {y_final}\n")
        print(f"Por lo tanto, la ecuación final tiene la forma: 1 = {alpha}({x_final}) + {n}({y_final})\n")

        # Convertimos de enteros negativos a enteros positivos módulo n, de presentarse el caso
        if x_final < 0:
            x_alt = -x_final
            x_alt = x_alt % n
            alpha_inv = n - x_alt
        else:
            alpha_inv = x_final % n
        # Comprobamos que alpha y alpha_inv verdaderamente sean inversos
        comp = (alpha * alpha_inv) % n
        if comp == 1:
            print(f"Comprobación: {alpha}({alpha_inv}) mod {n} = {comp}\n")-2]}\n")
```

Este código implementa el Algoritmo Extendido de Euclides para calcular el inverso multiplicativo de **alpha (α)** módulo **n**, y durante el proceso, también prepara ecuaciones para representar cada paso. El resultado final es el inverso multiplicativo y su verificación.

Pruebas solicitadas

1) $\boldsymbol{\alpha = 5}$, $\boldsymbol{\beta = 61}$, $\boldsymbol{n = 30}$ y después $\boldsymbol{\alpha = 17}$

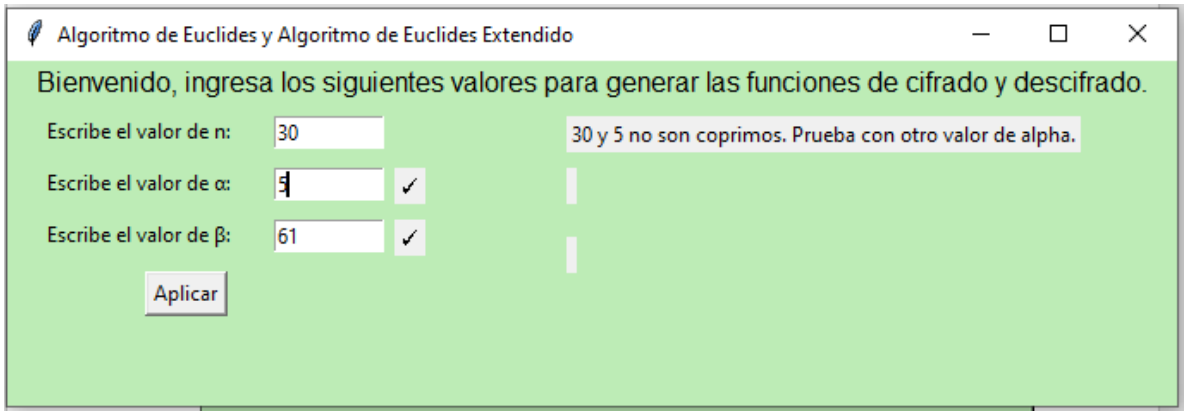

Imagen Prueba 1.- Evaluación de β menor a n

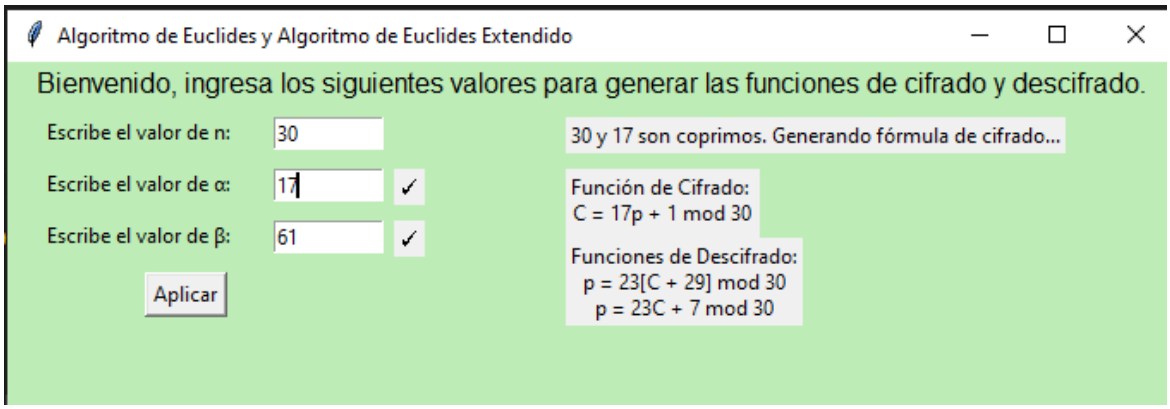

Imagen Prueba 1-2.- Evaluación con $\alpha = 17$

Como se observa en la imagen Prueba 1, $\boldsymbol{\beta}$ **es** mayor a **n**, entonces aplicamos modulo n a ese valor, para poder trabajar con ese valor. Al hacerlo, nos indica que 30 y 5 no son coprimos, por ellos cambiamos el valor de alfa a 17 (imagen Prueba 1-2) y ya podemos obtener las funciones de cifrado y descifrado.

2) $\boldsymbol{\alpha = 97, \beta = 12}$, $\boldsymbol{n = 239}$

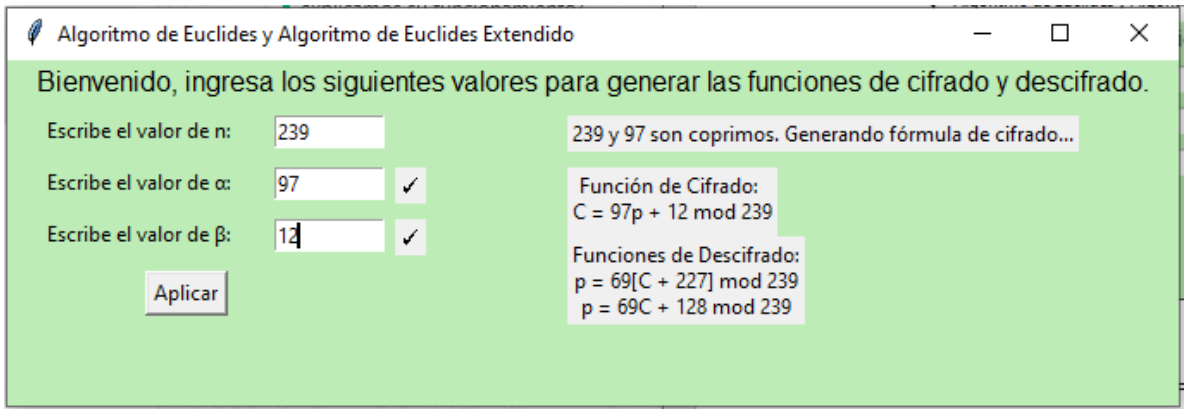

Imagen Prueba 2

Cuando se evalúa el valor de $\boldsymbol{\beta}$ y cumpla con la condición de que pertenezca al conjunto (0, n], entonces podemos procedes a aplicar el AE y AEE, donde **n** y $\boldsymbol{\alpha}$, son coprimos, como se observa en la imagen Prueba 2.

3) $\boldsymbol{\alpha = 11111, \beta = 1345}$, $\boldsymbol{n = 12345}$

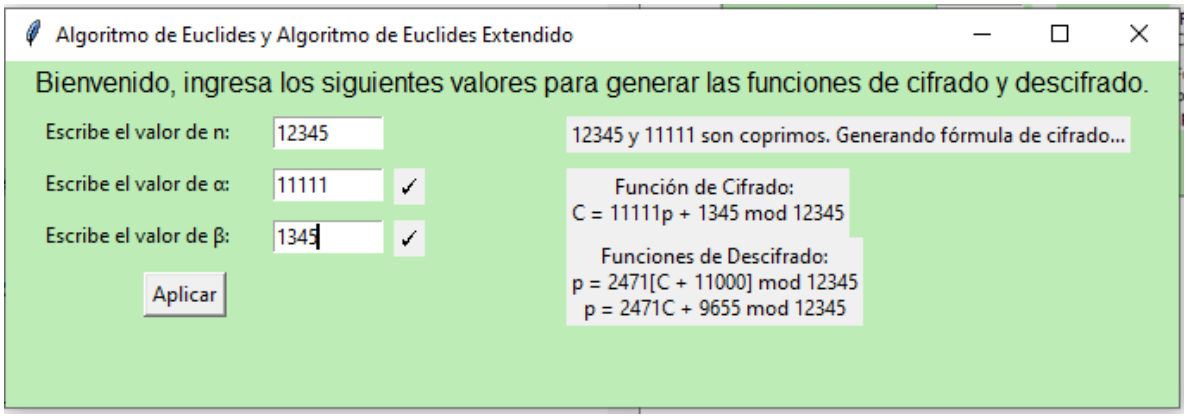

Imagen Prueba 3

Este programa permite trabajar con valores de cifras grandes y lo hace de manera eficiente a través del AEE, como se observa en la imagen Prueba 3 e imagen Prueba 5. Si quisiéramos encontrar, por ejemplo, el inverso multiplicativo a través de fuerza bruta, esto generaría muchísimas operaciones, pero el algoritmo lo hace a partir del procesamiento de funciones y es posible generar las funciones de descifrado más rápido y sencillo.

4) $\boldsymbol{\alpha = 13, \beta = 555}$, $\boldsymbol{n = 99991}$

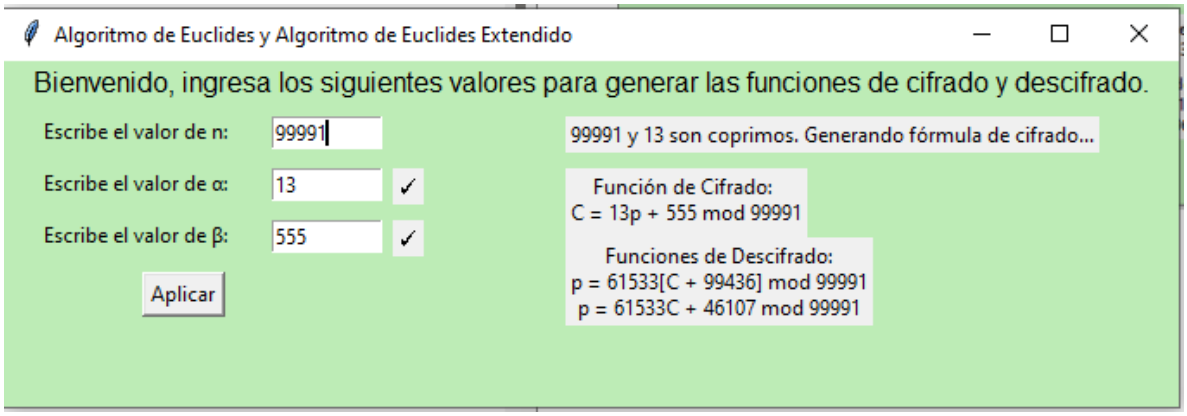

Imagen Prueba 4

Es recomendable trabajar con números primos como variables de entrada (visto en la imagen Prueba 4), para evitar que al momento de aplicar el AE para obtener el MCD, nos salga un número mayor a 1, y resulte imposible encontrar el inverso multiplicativo después.

5) $\boldsymbol{\alpha = 10009, \beta = 99999}$, $\boldsymbol{n = 104729}$

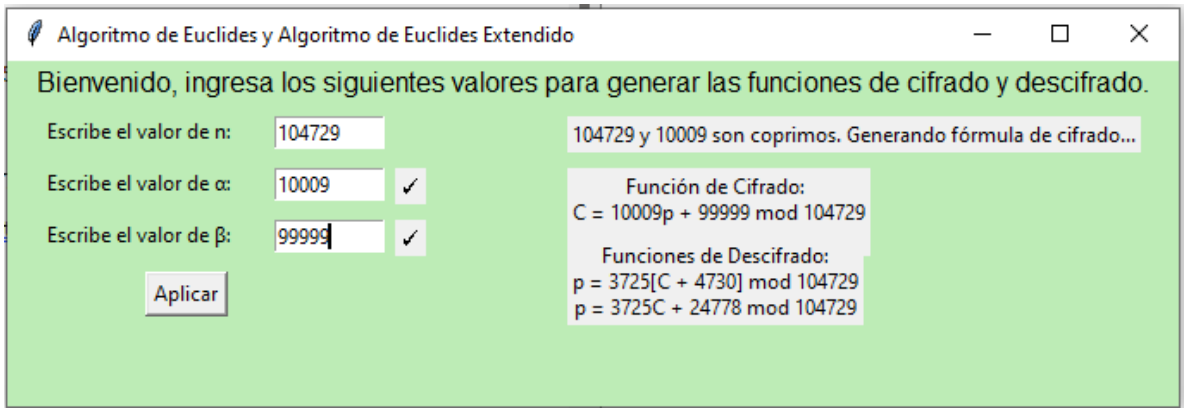

Imagen Prueba 5

Conclusión

Con base al desarrollo de la práctica y la implementación en Python de esta, logramos aplicar el funcionamiento de los Algoritmos de Euclides y Euclides Extendidos con la evaluación de los valores de $\boldsymbol{\beta}$, $\boldsymbol{\alpha}$ y **n**, donde si el valor de $\boldsymbol{\beta}$ era mayor al de **n**, aplicábamos modulo, sino no era posible comenzar con el funcionamiento del AE y tampoco del AEE, debido a que se ese valor de beta no existiría en el conjunto de n.

El proceso de desarrollo de ambos algoritmos inicialmente fue sencillo, específicamente con la parte de la implementación del AE, pues no resultó tan complejo manejar los cocientes y residuos para encontrar el MCD y para generar la función de cifrado. Por otra parte, tuvimos algunos detallitos con el AEE, más que nada por la abstracción de este, lo cual ocurrió en la parte del desarrollo de sustitución. Fue necesario recurrir a fuentes externas y fue así como encontramos acerca de la identidad de Bézout y sus coeficientes, cuyo cálculo existe gracias a una generalización del AEE. A partir de esto, logramos obtener los coeficientes x & y, y generar nuestras funciones de descifrado.

El criterio de selección de parámetros es muy importante durante el AE y AEE, pues depende de estos la funcionalidad del algoritmo y del programa entero. Si bien, anteriormente ya dijimos la importancia del valor de $\boldsymbol{\beta}$ con respecto a **n**, también debemos considerar que si tanto $\boldsymbol{\alpha}$ como $\boldsymbol{\beta}$ no son coprimos, no es posible aplicar el AEE, y tampoco podríamos sacar el inverso multiplicativo de nuestra función. Una sugerencia sería trata de trabajar con parámetros primos para proceder con la validación del MCD(n,α) = 1 y se pueda continuar con el funcionamiento del algoritmo.

El AEE resulta bastante eficaz para encontrar el inverso multiplicativo de algún número, independientemente de su tamaño. Si comparamos este algoritmo respecto al proceso de hacerlo por fuerza bruta, observamos un considerable ahorro de costo, debido a que la cantidad de operaciones por fuerza bruta es bastante grande respecto al tamaño de la cifra con la que trabajemos, cosa que no ocurre a menudo con el AEE.

Bibliografía

- *Algoritmo Extendido de Euclides (cálculo de los coeficientes de Bézout)*. (s. f.). ESFM. Recuperado 9 de octubre de 2023, de https://esfm.egormaximenko.com/linalg/gcd_extended_es.pdf

Criterios

Criterio	Valor	AE
El programa cuenta con interfaz gráfica que le permita al usuario introducir los parámetros	1	1
La función AE manda mensaje “prueba con otro valor” en caso de que α no sea coprimo con n	1	1
Una vez que se han validado α y β se muestran en la interfaz las funciones de cifrado y descifrado	1	1
La portada tiene: nombres completos, materia, nombre de profesor, fecha, logotipos, título de práctica y un resumen.	1	1
En media cuartilla con tus palabras explicar el AE y AEE, qué es y para qué sirve cada uno de ellos.	1	1
Código correspondiente a las dos funciones AE y AEE. (sugiero utilizar http://www.planetb.ca/syntax-highlight-word)	1	1
Captura de pantalla de las ejecuciones de las pruebas solicitadas. Todas las imágenes en el documento tienen título y se referencian en alguna parte del mismo. (Ej. “en la imagen 1 se muestra ...”)	1	1
Conclusiones en donde se exprese principalmente las dificultades de la implementación de la práctica (si es que las hubo) así como una reflexión sobre la selección de los parametros, sugerencias sobre el parametro multiplicativo. Comparación con el método de fuerza bruta para encontrar α^{-1}	1	1
Se incluyen referencias bibliográficas en formato IEEE o APA (teória o código)	1	1
TOTAL	9	9