Universidad del Valle de Guatemala
Facultad de Ciencias y Humanidades
Departamento de Matemáticas
Matemática Discreta
Ing. Mario Castillo
Sección 20

Proyecto:
Criptografía RSA

Andrea Amaya 19357
Brandon Hernández 19376
Laura Tamath 19365

Guatemala, Ciudad de Guatemala 25 de noviembre de 2020

Descripción y Metodología

Descripción

Para poder llevar a cabo este proyecto lo primero que se realizó fue conocer lo nuevo, se investigó del sistema de encriptación RSA. El cual consiste en dos fórmulas para poder encriptar y desencriptar:

- Encriptar: $m^e \% n = c$
- Desencriptar: $c^d \% n = m$

En donde ***e*** y ***n*** forman la llave pública, ***d*** forma la llave privada, ***m*** es el mensaje desencriptado y ***c*** representa al mensaje encriptado. Para poder obtener los valores de *e*, *n* y *d* es necesario seguir los siguientes pasos:

1. Definir dos números primos arbitrarios muy grandes ***p*** y ***q***.
2. Calcular **phi**, el cual es una constante que se basa en los dos números primos.
3. Calcular los valores de ***e*** y ***d***.

En el primer paso se escogen dos valores para las dos variables, se recomienda que ambas sean de un tamaño de alrededor de 2048 bits, ya que son la base de las llaves. Continuamos por calcular el valor de phi, el cual es obtenido por la siguiente expresión:

$$\varphi = (p - 1)(q - 1)$$

Al momento de calcular esto es importante que **phi** y ***e*** no compartan algún factor, de lo contrario no ***e*** no poseerá inverso modular. Se prosigue asignándole un valor arbitrario a la variable ***e***, que normalmente se le asigna el valor de tres. Finalizando el proceso, es necesario encontrar el inverso de ***e*** que es ***d***, para cumplir con la siguiente expresión:

$$(e * d) \% \varphi = 1$$

(Vance, 2014)

Para poder calcular el inverso modular de manera sencilla se utiliza el ***Algoritmo de Euclides***, el cual es un método eficiente para calcular el máximo común divisor entre dos números. Por ejemplo, si se desea encontrar el MCD de dos números x_1 y x_2 el algoritmo indica que:

- Si $x_1 = 0$, entonces el MCD de (x_1, x_2), es x_2, debido el entero más grande que puede dividir uniformemente a x_2 es x_2, y todos los enteros (C) dividen uniformemente a 0, lo cual se puede expresar como $C \cdot 0 = 0$. Así se puede concluir que x_2 debe dividir uniformemente a 0, por ende, el número más grande que divide tanto a x_2 como a 0, es x_2.
- Si $x_2 = 0$, entonces el MCD de (x_1,x_2), es x_1, por lo mismo que se indica anteriormente, solo que el número más grande que divide tanto a x_1 como a 0, es x_1 .
- Si $x_1 \neq 0$ y $x_2 \neq 0$, entonces se utiliza división larga para encontrar que $x_1/x_2 = A$, con residuo R, donde A es un entero y R es un entero entre 0 y x_2 - 1.

Esto se puede escribir como: $x_1 = x_2 \cdot A + R$. Por ende, se encuentra el MCD(x_2,R) al usar el algoritmo de Euclides, ya que MCD(x_1,x_2) = MCD(x_2,R).

(Khan Academy, S.F)

Implementación

func.py

- Se hace uso de las librería random para generar números primos aleatorios
- La función **encrypt** se encarga de:
    - Recibir el mensaje encriptado
    - Generar un dos números primos (**p, q**) con la función generatePrime(rango1, rango2)
    - Calcular la multiplicación entre p y q (**n**)
    - Calcular el mcm (**phi**) con la función lcm(p-1, q-1)
    - Calcular la llave pública (**e**) con la función publicKey()
    - Calcular (mensaje**e)%n con la función ciphertext(mensaje, e, n)
    - Se retorna el mensaje encriptado
- La función **publicKey** se encarga de:
    - Generar la llave pública
    - Retornar la llave pública 65537
- La función **privateKey** se encarga de:
    - Generar la llave privada haciendo uso de **e** y **phi**
    - Calcular el modular inverso haciendo uso de la función mod_Inv(e, lcm)
    - Se retorna la llave privada
- La función **decrypt** se encarga de:
    - Recibir la llave privada
    - Calcular (c**privateKey)%n
    - Se retorna el mensaje descifrado

Interfaz Gráfica:

- Se utiliza la librería Tkinter la cual es un binding de la biblioteca gráfica Tcl/Tk
- Librería Os para crear un archivo dentro de una carpeta
- Se importa la librería PIL la cual permite la edición de imágenes directamente desde Python
- Se importan las funciones de func.py
- Se define que el tamaño de la ventana interfaz sería de 500 x 455 píxeles
- Se añade el nombre de la ventana “Proyecto | Matemática discreta”
- Se agrega el título “criptografía RSA”
- El botón **llave pública**, ejecuta la función *writePrivateKey* encargada de:
    - Revisar que exista el documento encrypt y lo abre
    - Verificar que el documento donde se encuentran las llaves exista y lo abre

    - Mostrar un mensaje en el cual indique que se ha generado la llave privada
    - Si alguno de estos dos archivos no son encontrados, entonces muestra un mensaje de error
- El botón **Encriptar** se ejecuta la función *encryptTxt* la cual es encargada de:
    - Si se generó la llave pública, entonces:
        - Abre el documento que posee el texto a encriptar
        - El contenido se guarda en una variable llamada *lines*
        - Elimina los .txt que se encontraban en encrypt
        - Crea un nuevo .txt encrypt
        - Si no hay ningún .txt que eliminar, entonces crea un nuevo encrypt.txt
        - Encripta el texto guardado en la variable *lines*
        - De una lista de enteros, se traslada a una lista de Strings
        - Muestra un mensaje que indica si se generó la encriptación
    - Muestra un mensaje de error indicando que la llave pública no se ha generado
- El botón **Llave privada** se ejecuta la función *writePrivateKey* la cual se encarga de:
    - Revisar si existe el texto encriptado
    - Abrir el archivo .txt que posee las llaves
    - Escribe la llave privada
    - Muestra un mensaje indicando que la llave privada se ha generado
    - Si no existe el archivo encriptado ni el texto con las llaves, muestra un mensaje indicando que no hay nada encriptado.
- El botón **Desencriptar** se ejecuta la función *writeDecrypt* la cual es encargada de:
    - Si se generó la llave privada, entonces:
        - Elimina los .txt que se encontraban en decrypt
        - Crea un nuevo .txt decrypt
        - Si no hay ningún .txt que eliminar, entonces crea un nuevo encrypt.txt
        - Muestra un mensaje indicando que se ha generado el desencriptado
    - Si no se ha generado la llave privada, muestra un mensaje indicando que aún no hay llave privada
- La función **closing** se utiliza para preguntarle al usuario si está seguro de querer cerrar el programa.

Manual de Usuario
https://www.canva.com/design/DAENKpNlpFY/chmdtMBalyYqsUJGjH37Eg/view?utm_content=DAENKpNlpFY&utm_campaign=designshare&utm_medium=link&utm_source=sharebutton

Literatura Citada

- Lake, J. (10 de diciembre de 2018). *What is RSA encryption and how does it work?*. Comparitech. https://www.comparitech.com/blog/information-security/rsa-encryption/
- Vance, A. (14 de octubre de 2014). *How the RSA algorithm works, including how to select d, e, n, p, q and φ (phi)*. Youtube. https://www.youtube.com/watch?v=Z8M2BTscoD4
- Khan Academy. (S.F). *Inversos modulares*. KhanAcademy.: https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/modular-inverses