UNIVERSIDAD NACIONAL AUTÓNOMA DE NICARAGUA, LEÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

DEPARTAMENTO DE MATEMÁTICA, ESTADÍSTICA Y ACTUARIALES

MONOGRAFÍA PARA OPTAR AL TÍTULO DE LICENCIATURA EN MATEMÁTICA

ANÁLISIS DE LA EFICIENCIA DEL ALGORITMO DE CÉSAR Y RSA PARA ENCRIPTAR MENSAJE DE TEXTO

ELABORADO POR:

Br. JOADAN HUZIM DUARTE RIZO

Br. RANDY JOSÉ MAKIS COLEMAN

Br.JOHAXEL ANTONIO TÓRREZ MARTÍNEZ

TUTORA:

M.Sc. LISSETTE DEL CARMEN QUINTERO VARGAS

CO-TUTOR:

M.Sc. JUAN CARLOS LEYTÓN BRIONES

LEÓN, 18 DE AGOSTO DE 2023

“ A LA LIBERTAD POR LA UNIVERSIDAD ”

DEDICATORIA

Primeramente esta tesis está dedicada a Dios, ya que gracias a él hemos logrado concluir nuestra carrera Universitaria, a nuestros padres porque ellos siempre han estado brindándonos apoyo para hacer una gran persona y a nuestros respetados maestros, a quienes consideramos nuestros mentores, por brindarnos sus conocimientos compartidos y por su pasión por la enseñanza, gracias a ustedes hemos adquirido las herramientas necesarias para abordar los desafíos de este trabajo de investigación.

Esta tesis es el resultado del arduo trabajo, esfuerzo y compromiso de todos aquellos que han confiado en nosotros. Su apoyo inquebrantable y su influencia positiva han sido fundamentales para mi formación académica y personal.

Les dedicamos este logro, esperando que nuestro trabajo contribuya de alguna manera al avance del conocimiento y al bienestar de la sociedad.

AGRADECIMIENTOS

El principal agradecimiento es a Dios quien nos ha guiado y nos ha dado fortaleza para seguir adelante. A mi familia por su comprensión y estímulo constante, además su apoyo incondicional a lo largo de nuestro estudio y también a los maestros que hicieron parte de este proceso de integral formación, que deja como producto terminado este grupo de graduados; como recuerdo y prueba viviente en la historia, esta tesis que perdurará dentro de los conocimientos y desarrollo de las demás generaciones que esta por llegar.

Finalmente agradecemos a quien lee este apartado y más de nuestra tesis, por permitirnos experiencia y nuevos conocimiento.

RESUMEN

En el siguiente trabajo de investigación se presenta un análisis exhaustivo de la eficiencia de dos grades algoritmos como lo son el algoritmo de Cesar el cual lleva el nombre gracias a Julio Cesar que inventó un sistema criptográfico que cuya función es que cada letra del mensaje se desplaces a un número fijo de la posición de alfabeto y El RSA es un algoritmo de cifrado de clave pública ampliamente utilizado en seguridad de la información. En términos de capacidad de encriptación, el algoritmo RSA ofrece un alto nivel de seguridad, ya que se basa en la dificultad de factorizar números enteros grandes y así poder comparar la eficiencia que tiene cada algoritmo para poder proteger un mensaje encriptado. De tal manera ambos algoritmos fueron ejecutados en el lenguaje de programación C, donde se aplicó la teoría de números para establecer las condiciones que ambos algoritmos utilizan para poder encriptar y desencriptar mensajes de texto.

ÍNDICE GENERAL

CAPÍTULO 1

ASPECTOS INTRODUCTORIOS

1.1. Antecedentes

En la revista de espacio Cuzco, Mantanilla, Mendez, y avila (2019) se estudio "Experiencia de aplicación de criptografía para mejorar la seguridad de un método esteganográfico en imágenes" El algoritmo criptográfico existentes se seleccionó el cifrado de César por su gran compatibilidad con este, pues en las pruebas realizadas se observó que no hay cambios visibles en la imagen.

Según Rinza, Sayago, y Rocha-Rinza (2020)"descifrado de César por medio de análisis de frecuencia para tres idiomas". Cifrar por medio del algoritmo de César y se descifrar por fuerza bruta, mediante un análisis de frecuencias, se introduce el análisis de frecuencia para 3 idiomas para que el criptograma, se pueda descifrar en estos 3 idiomas elegidos, los textos llanos pueden estar en español, inglés, y portugués, la interfaz contará con tres secciones (texto plano, texto cifrado y texto descifrado), un área de salida de datos tipo consola y un área de comandos. Estos deben permitir las acciones de cargar texto desde un archivo, limpiar los campos, realizar el cifrado y descifrado, seleccionar un idioma de trabajo y realizar el ataque por análisis de frecuencias. Ya que el análisis de frecuencia varía dependiendo del idioma que utilicemos.

Universidad Nacional de Ucayali Tominaga García (2022). Donde se estudio el "sistema cifrado roce en la seguridad de archivos de texto". se planteo una aplicación de lo sistema de encriptamiento de texto usando el algoritmo de roce.

1.2. Introducción

Tanto en el campo de la matemática como en la seguridad, la protección de datos sensibles y la comunicación confidencial han sido un tema de gran importancia donde dos de estos métodos criptográficos conocidos como el algoritmos César y RSA donde a través de la teoría de números damos función a estos algoritmos.

El cifrado César es uno de los métodos de cifrado más simples y antiguos. Opera mediante el desplazamiento de cada letra en un mensaje un número fijo de posiciones en el alfabeto. Aunque es fácil de entender e implementar (solo 26 posibles desplazamientos) y la falta de distribución de probabilidad realista en las letras del lenguaje. En términos de eficiencia, el cifrado César es extremadamente rápido, ya que solo implica operaciones de suma y módulo.

Por otro lado, el cifrado RSA es un algoritmo de criptografía asimétrica que se basa en la dificultad de factorizar grandes números enteros en números primos.

Estos juegan un papel crucial en la parte de la criptografía donde nos garantiza la privacidad y la integridad de la información en diversas aplicaciones, pero ¿Cómo podemos comprobar la eficiencia de este sistema de encriptación del algoritmo de César y RSA? ¿Como se puede desarrollar un criptosistema en el lenguaje c, utilizando el algoritmo de César y RSA?.

1.3. Objetivos

Objetivo General

Analizar la eficiencia de algoritmo de cifrado de César y RSA para Encriptar mensaje de texto.

Objetivos Especifico

- Aplicar la congruencia modular en el algoritmo de César y RSA para poder cifrar texto.
- Desarrollar una implementación del algoritmo de cifrado de César Y RSA en el lenguaje de programación c.
- Demostrar el uso practico de algoritmo de César y RSA como una manera de Encriptación para mantener los mensaje de texto cifrado y que se pueda decifrar.
- Comparar la eficiencia del algoritmo de César y RSA para el encriptamiento de mensaje de texto.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Aritmética modular

Según Grimaldi (1994) ahora se resaltara algunas propiedades básica , definiciones y teorema de la aritmética modular que explica:

Definición 2.1.1. *Si a y b son enteros y m es un entero positivo, entonces a es congruentemente con b módulo m si divide a a − b. Usamos la notación* $a \equiv b(mod\ m)$ *para indicar que a es congruente con b módulo m. Si a y b no son congruentes módulo m, escribimos* $a \not\equiv b(mod\ m)$.

Teorema 2.1.1. *Sea a y b numero enteros y m un entero positivo. Entonces* $a \equiv b\ (mod\, m)$ *si, solo si* $a\ mod\ m = b\ mod\ m$

Demostración: Sin perdida de generalidad como $a \equiv b\ (mod\, m)$ entonces por definición de modulo tenemos $m|(a-b)$ lo que tenemos que probar es que $a\ mod\ m = b\ mod\ m$, definamos $d_1 = mcd(a, m)$ entonces

$$d_1|a$$

$$d_1|m$$

pero como $m|(a-b)$ esto implica que

$$d_1|(a-b)$$

por tanto $d_1|\,[a-(a-b)]=b$, de manera reciproca garantizamos $d_1|b$ y $d_1|m$ esto prueba que $d_1|d_2$

ahora como $d_2=mcd(b,m)$ esto implica que $d_2|b$ y $d_2|m$ entonces

como m divide $a-b$ $d_2|(a-b)$ por tanto

$$d_2|\,[b+(a-b)]=a$$

asi que $d_2|a$ y $d_2|m$ esto prueba $d_2|d_1$ por tanto demostramos que $a\ mod\ m=b\ mod\ m$. ■

Teorema 2.1.2. *Sea m un entero positivo. Los enteros a y b son congruentes módulo m si, y sólo si, existe un entero k tal que $a=b+km$.*

Demostración: Si $a\equiv b\,(mod\ m)$, entonces $m|\,(a-b)$. Esto significa que hay un entero K tal que $a-b=Km$, por lo que $a=b+Km$. Recíprocamente, si hay un entero K tal que $a=b+Km$, entonces $Km=a-b$. Así, m divide a $a-b$, por lo que $a\equiv b\,(mod\ m)$. ■

Teorema 2.1.3. *Sea m un entero positivo. Si $a\equiv b(mod\ m)$ y $c\equiv d(mod\ m)$, entonces $a+c\equiv b+d(mod\ m)$ y $ac\equiv bd(mod\ m)$.*

Demostración: Como $a\equiv b\,(mod\ m)$ y $c\equiv d\,(mod\ m)$, existirán dos enteros s y t tales que $b=a+sm$ y $d=c+tm$. Por lo tanto.

$b+d=(a+sm)+(c+tm)=(a+c)+m\,(s+t)$

entonces

$bd=(a+sm)\,(c+tm)=ac+m\,(at+cs+smt)$

Asi.

$a+c\equiv b+d\,(modm)\,yac\equiv bd\,(modm)\,.$ ■

2.2. Algoritmo Euclidianos

De acuerdo con Grimaldi (1994) donde garantiza alguna definiciones de algoritmo Euclidianos y tenemos en esta sección algunos lemas y teoremas del máximos común

divisor y congruencia lineal que explican:

Definición 2.2.1. *Para $a, b \in \mathbb{Z}$ Un entero positivo c es un común divisor de a y b si $c|a$ y $c|b$.*

Definición 2.2.2. *Sean $a, b \in \mathbb{Z}$ Donde $a \neq 0$ o $b \neq 0$. Entonces $c \in \mathbb{Z}^+$ es el maximo común divisor de a y b si*

1. *$c|a$,$c|b$ (es decir, c es un divisor común de a, b)*

2. *Para cualquier común divisor d de a y b tenemos $d|c$.*

2.2.1. Máximos común divisor

Lema 2.2.1. *Sea $a = bq+r$, donde a, b, q y r son enteros. Entonces, $mcd\,(a, b) = mcd\,(b, r)$.*

Demostración: Si podemos demostrar que los divisores comunes de (a y b) son los mismos que los comunes divisores de (b y r), habremos demostrado que $mcd\,(a, b) = mcd\,(b, r)$, puesto que ambas parejas deben tener el mismo mayor divisor común.

■

Teorema 2.2.1. *Sea m un entero positivo y sean a, b y c enteros. Si $ac \equiv bc\,(mod\ m)$ y $mcd\,(c, m) = 1$ entonces $a \equiv b\,(mod\ m)$.*

Demostración: Como $ac \equiv bc\,(mod\ m)$, $m|ac - bc = a\,(a - b)$ según el Lema 1. como $cmd\,(c, m) = 1$, se sigue que $m|\,(a - b)$.Se concluye, por tanto, que $a \equiv b\,(modm)$.

■

2.2.2. Congruencia lineal

Lema 2.2.2. *Si a, b y c son enteros positivos, tal que $mcd\,(a, b) = 1$ y $a|bc$, entonces $a|c$.*

Demostración: Como , $mcd\,(a, b) = 1$ por el Teorema de máximo común divisor hay dos enteros s y t tales que $sa + tb = 1$.

Multiplicando ambos lados de la ecuación por c obtenemos $sac + tbc = c$

Usando el teorema básico de divisibilidad de entero, podemos utilizar la ecuación anterior para mostrar que $a|c$. Según la parte $a|tbc$. Como $a|c$ y $a|tbc$. por el enciso a del teorema se concluye que a divide a $ac + tbc$, y por lo tanto $a|c$.

■

Teorema 2.2.2. *Si a y m son primos relativos y $m > 1$, entonces existe un inverso de a módulo m. Ademas, este inverso es único módulo m. Esto es, hay un unico entero positivo $\overline{a}$ menor que m que es inverso de a módulo m y cualquier otro inverso de a módulo m es congruente con $\overline{a}$ módulo m.*

Demostración: Según el teorema básico de divisibilidad de entero, como $mcd\,(a, m) = 1$, hay dos entero s y t tales que $sa + tm = 1$.

Esto implica que $sa + tm = 1\,(mod\ m)$.

Como $tm \equiv 0\,(mod\ m)$, se sigue que

$sa \equiv 1\,(mod\ m)$.

Por tanto, s es un inverso de a módulo m.

■

2.2.3. Criterio de congruencia

Según Grimaldi (1994) se considera que los métodos de los criterios de congruencia que están definidos por:

Dado $a, b \in \mathbb{Z}$ y un $n \in \mathbb{Z}$, $n \geq 0$ fijo. Diremos que a es congruente con b modulo m, o que están en la relación de congruencia módulo m, y se indica por $a \equiv b(mod\ m)$ si y solo si $\dfrac{m}{(a-b)}$. Las dos definiciones siguientes son equivalentes:

- $a \equiv b(mod\ m)$ si y solo si $n = (a - b)$ a y b dejan el mismo residuo al dividirse entre n.
- $a = nq = b$ se le llama modulo de la congruencia, que no es más que el número entero positivo que tiene el papel de divisor en la definición de congruencia.

Teorema 2.2.3. *La relación de congruencia se puede reescribir como: $a \equiv b\ (mod\ m)$ si solo si el resto de la división euclídea de a y de b por n es el mismo.*

Demostración: Supongamos primero que $a \equiv b\ (mod\ m)$

$a \equiv b\ (mod\ m) \Longrightarrow$ Existe $k \in \mathbb{Z}$ tal que $a - b = km$. Al realizar la división euclídea de b por n se tiene: $b = pn + r$ con $0 \leq r < m$. sustituyendo b en la expresión anterior se tiene que $a = (k + p)m$, con $0 \leq r < m$. se ha obtenido que el resto de la división euclidiana de a por m es también r.

Reciprocamente, supongamos el resto de la división euclidiana de a y b por m es el mismo $a = qm + r$ y $b = pm + r$ con $0 \leq r < m$

Restando se obtiene $(a-b) = (q-p)m$ donde $a-b$ es un múltiplo de m, lo que significa que $a \equiv b\ mod\ m$. ■

2.3. Teorema Chino

Teorema 2.3.1. ***Teorema chino*** *Hellman (1976) Sean $m_1, m_2, \cdots, m_n$ enteros positivos primos relativos dos a dos. El sistema*

$$x \equiv a_1\ mod\ m_1, \equiv a_2\ mod\ m_2, \cdots, \equiv a_n\ mod\ m_n$$

tiene solución única módulo $m = m_1, m_2, \cdots, m_n$

Demostración: Demostraremos que esta solución existe describiendo una forma de construirla. Se verá que esta solución es única módulo m.

Para construir una solución del sistema, primero consideramos

$$M_k = m|m_k$$

para $k = 1, 2, ..., n$, Esto es, M_k es el producto de lo modulo exceptuando m_k. como m_i y m_k no tiene factores mayores que 1 cuando $i \neq k$ se sigue lo siguiente que $mcd(m_i, m_k) = 1$. Por lo tanto, haciendo uso de teorema de lo primos relativo consideramos que existe un entero y_k que es el inverso de M_k modulo m_k, esto es,

$$M_k y_k \equiv \ mod\ m_k$$

Para construir una solución a toda la ecuación, es considerando una suma

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + ... + a_n M_n y_n$$

Ahora veremos que x es una solución del sistema de congruencias. Primero, observa que como $M_j \equiv 0\ (mod\ m_k)$ siempre que $j \neq k$,todos los términos de esta suma excepto el k-ésimo son congruentes con O módulo m_k, como $m_k y_k \equiv 1\ (mod m_k)$ vemos que

$$x \equiv a_k M_k y_k \equiv a_k\ (mod\ m_k)$$

Para $k = 1, 2, ..., n$. Se ha demostrado que x es una solución simultánea de las n congruencias. ■

2.4. Introducción a la Criptografía

De acuerdo Paredes (2006) las aplicaciones actuales más interesantes de la Aritmética Modular ocurre en la Criptografía, en la que se utilizan las distintas operaciones, con el objeto de cifrar información. La definición de los distintos sistemas deciframiento, como se verá en esta sección, es suficientemente simple para ser abordados por cualquier persona con unos conocimientos básicos de la Aritmética Modular.

En esta sección describiremos cómo se pueden usar algunos conceptos básicos de Aritmética Modular en el ciframiento de datos, para ello presentamos a continuación la definición de un sistema criptogría o Criptosistema. Un sistema criptográfico o criptosistema S es una sextupla.

2.4.1. Sistema Criptográfico

Según Moreno y Díaz (2020) Ahora se resaltarán los algoritmos matemáticos y se describirá las operaciones de cifrado y descifrado de cada método de encriptación.

2.4.2. Cifrado de César

El primer método para estudiar es el Cifrado de Julio César que le debe su nombre al emperador Julio César quien inventó un sencillo método para mantener algunos de sus escritos en secreto y para comunicarse con los generales en las batallas, aunque es uno de los más simples en cuanto al proceso de cifrado para aquellos tiempos no era tan común que las personas supieran leer por lo cual tuvo gran efectividad en un comienzo, pasando a la historia como el Cifrado del César.

Este tipo de cifrado se conoce con el nombre de cifrados monográficos, es decir aquellos que están basados en la sustitución de cada símbolo del alfabeto por otro (Lucena, 2001). Para este método el proceso de cifrado consiste en sustituir cada letra por la que se encuentra tres posiciones más adelante en el orden alfabético, para el alfabeto castellano formado por 27 letras donde se excluye las letras CH y LL quedaría de la siguiente manera para cifrar un mensaje claro.

2.4.3. Algoritmo Matemático del cifrado del César

De acuerdo Fernandez (2004) explica:

Definición 2.4.1. *Sean* $K = \mathbb{Z}_{27}$*, para* $0 \leq 26$*, se define:*

1. $e_k(x) = x + K \ mod\ 27$ *para el ciframiento*

2. $e_k(x) = x - K \ mod\ 27$ *para el deciframiento*

donde x *y* y *perteneces* $\mathbb{Z}_{27}$.

2.4.4. Sistema RSA

En esta sección se va a describir un sistema criptográfico que se conoce como RSA. La idea es transmitir mensajes por canales “inseguros” (esto es, accesibles a individuos distintos del emisor y del receptor) sin que puedan ser comprendidos más que por el emisor y el receptor. Esto exige un proceso de codificación del mensaje y su posterior decodificación. Los caracteres del mensaje se traducen a números, se envían números.

2.4.5. Teorema Euler.

Dado un número $N = p_1^{a_1}, p_2^{a_2}, ..., p_n^{a_n}$ son números primos, hay números $\phi(N)$ exactos entre 1 y $(N-1)$ que son coprimos de N, Donde:

$$\phi(N) = \prod_{i=1}^{n}(p_n^{a_n-1}, (p_n - 1)) \text{ Paillier (1996)}$$

2.4.6. Algoritmo matemático RSA

Caso RSA

Codificación:

1. Se eligen dos números primos grandes p y q y se considera n= $p.q$ donde

   p es un gran número primo desconocido y $p = p_1 + p_2 + ... + p_n = \sum_{i=1}^{n} p_i$

   q es un gran número primo desconocido y $q = q_1 + q_2 + ... + q_n = \sum_{i=1}^{n} q_i$

2. Los componentes públicos son módulos N y el exponente de cifrado e donde e es coprimo con N donde $1 \leq e < (p-1)(q-1)$,MCD$(e, (p-1)(q-1)) = 1$ por el teorema de euler

3. Se transforma el número entero M, que representa el mensaje a enviar, en C donde $C \equiv M^e \ (mod\ m)$ Hellman (1976)

Descifrado:

El siguiente teorema, que demostraremos más adelante, justifica el descifrado.

1. Pequeño teorema de Fermat: "Si p es primo y a es un entero no divisible por p, entonces $a^{m-1} \equiv 1(mod\ m)$"

2. El mensaje se puede recuperar cuando se conoce la clave de descifrado

   El exponente de descifrado desconocido $d = d_1 + d_2 + ... + d_n = \sum_{i=1}^{n} d_i$

3. Los componentes privados son factores primos de N, que son p y q, y el exponente de descifrado d donde: $de \equiv 1\ mod\ (p-1)(q-1)$, sin perdida de generalidad

$C^d \equiv (M^e)^d \ mod\ m = M^{ed} \equiv M^{1+k(p-1)(q-1))} \ mod\ m \equiv M(M^{(p-1)})^{k(q-1)} \ mod\ m$ entonces se verifica que $n = p \cdot q$ tenemos lo siguiente

$$C^d \equiv (M^{(p-1)})^{k(q-1)} \ mod\ m \equiv M \cdot 1 \ mod\ p$$
$$C^d \equiv (M^{(q-1)})^{k(p-1)} \ mod\ m \equiv M \cdot 1 \ mod\ q$$

Ahora al saber que C^d son solución del sistema de congruencias entonces $x \equiv M \ (mod\ p), x \equiv M \ (mod\ q)$ aplicando el teorema Chino el restos se sigue que la solución, M, s única módulo $MCD(pq) = p \cdot q = n$

por tanto $C \equiv M^e \ (mod\ m)$.

■

2.4.7. Criptosistema de rabín

El criptosistema de rabín es una aplicación de la teoría de residuo cuadráticos y el teorema chino de los residuos en la criptografía,

2.4.8. Algoritmo matemático del cifrado de rabín

Definición 2.4.2. *sea $n = p \cdot q$ en donde p.q son primos y $p \cdot q \equiv 3 \ \ mod\ 4$ entonces* $P = C = \mathbb{Z}_n$, $k = n, p, q$ *para* $k = (n, p, q)$

$e_k(x) = x^2 \ mod\ n$

$d_k(y) = \sqrt{y} \ \ mod\ n$ *ElGamal (1985)*

2.5. Criptosistemas Simétricos

Según Paredes (2006) los sistemas de cifrado simétrico son aquellos que utilizan la misma clave para cifrar y descrifrar un documento. El principal problema de seguridad

reside en el intercambio de claves entre el emisor y el receptor ya que ambos deben usar la misma clave. Por lo tanto se tiene que buscar también un canal de comunicación que sea seguro para el intercambio de la clave. Es importante que dicha clave sea muy difícil de adivinar ya que hoy en día los ordenadores pueden adivinar claves muy rápidamente. Debemos tener en cuenta que los algoritmos criptográficos son públicos, por lo que su fortaleza debe depender de su complejidad interna y de la longitud de la clave empleada para evitar los ataques de fuerza bruta.

Por ejemplo el algoritmo de cifrado DES usa una clave de 56 bits, lo que significa que hay 72 mil billones de claves posibles. Actualmente ya existen ordenadores especializados que son capaces de probar todas ellas en cuestión de horas. Hoy por hoy se están utilizando ya claves de 128 bits que aumentan el "espectro" de claves posibles (2^{128}) de forma que aunque se uniesen todos los ordenadores existentes en estos momentos no lo conseguirían en miles de millones de años.

El funcionamiento de la criptografía simétrica es el siguiente: el emisor quiere hacer llegar un documento al receptor. Toma ese documento y le aplica el algoritmo simétrico, usando la clave única que también conoce el receptor. El resultado es un documento cifrado que se puede ya enviar tranquilamente. Cuando el receptor recibe este documento cifrado, le aplica el mismo algoritmo con la misma clave, pero ahora en función de descifrar. Si el documento cifrado no ha sido alterado por el camino y la clave es la misma, se obtendrá el documento original.

Las principales desventajas de los métodos simétricos son la distribución de las claves, el peligro de que muchas personas deban conocer una misma clave y la dificultad de almacenar y proteger muchas claves diferentes. La seguridad en clave simétrica reside en la propia clave secreta, y por tanto el principal problema es la distribución de esta clave a los distintos usuarios para cifrar y descifrar la información. La misión del emisor y receptor es mantener la clave en secreto. Si cae en manos equivocadas ya no podríamos considerar que la comunicación es segura y deberíamos generar una nueva clave. Para superar estas desventajas que presentaba el sistema de criptografía de clave privada se desarrolló en 1976 el sistema de criptografía asimétrica o de clave pública.

2.5.1. Criptografía Moderna

Los criptosistemas moderno, cuya cifra en bits está orientado a todos los carácter del codigo ASCII o ANCI usan por lo general una operación algebraica en $\mathbb{Z}_n$, un cuerpo finito, sin que necesariamente este módulo deba corresponder con el número de elemento del alfabeto o código utilizado. Paredes (2006)

2.5.2. Algoritmos de Clave Secreta Compartida

Para estos algoritmos, lo primordial es el intercambio de las claves, debe ser muy cuidadoso para que las claves no caiganen manos equivocadas.

Un protocolo de establecimiento de claves, también llamados protocolos de intercambio de claves es un protocolo criptográfico en el que se establece una secuencia de pasos entre dos o más participantes a través de la cual los participantes se ponen de acuerdo en el valor de una información secreta compartida. A la información secreta compartida se le suele llamar clave debido a que esa información se suele usar como clave de algún algoritmo criptográfico.test. Moreno y Díaz (2020)

2.5.3. Criptoanálisis

"Aplicado a la ciencia se dice que incluye todos los principios, métodos y medios empleados en el análisis de criptogramas, es decir, su reducción o solución sin el conocimiento del sistema o la clave, o la posesión del libro de códigos, mediante un detallado estudio de los criptogramas en sí. Criptoanálisis es el nombre aplicado a los pasos ejecutados en la aplicación de los principios de la criptoanálisis a los criptogramas. Un criptoanálisis es un experto en la aplicación de las operaciones o procesos en el criptoanálisis.

- Análisis de consumo: El tercer avance es el análisis del consumo de energía eléctrica para averiguar las claves secretas. Las computadoras por lo general utilizan 3 voltios para representar un bit 1 y 0 voltios para representar un bit 0. Por lo tanto, procesar un 1 gasta más energía eléctrica que procesar un 0.

- Análisis de temporización: Los algoritmos criptográficos están llenos de instrucciones if que prueban bits en las claves de ronda. Si las partes then y else toman diferentes cantidades de tiempo, reduciendo la velocidad del reloj y viendo el tiempo que tardan en ejecutarse varios pasos, también podría ser posible deducir las claves de ronda. Moreno y Díaz (2020)

2.6. Firma Digitales

- Codificación de mensaje : Es cuando se combinan dos funciones dispares: autenticación y confidencialidad. Se base en la idea de una función de hash unidireccional que toma una parte arbitrariamente grande de texto llano y a partir de ella calcula una cadena de bits de longitud fija. Tiene 4 propiedades importantes:

    1. Dado P facil calular MCD(P).
    2. Dado MCD(P), es imposible encontrar P.
    3. Dado P nadie puede encontrar P' de tal manera $MCD(P')MCD(P)$.
    4. Un cambio de entrada de incluso 1 bit produces. Moreno y Díaz (2020)

CAPÍTULO 3

DISEÑO METODOLÓGICO

3.1. Tipo de estudio

La investigación es de tipo Cuantitativo , pues consiste en la optimización y análisis de la eficiencia de seguridad que tiene el algoritmo de César para poder encriptar un mensaje de texto , como problema tipo práctico y bansandonos también en la teoría de número para poder cumplir con lo objetivos propuesto se seguirá el siguiente esquema en 4 etapas :

- Estudio.
- Selección de la herramienta a usar.
- Desarrollar un algoritmo para poder encriptar mensaje de texto.
- Evaluación y valoración.

Etapa I. Estudio

En esta etapa se realizará completamente para obtener el algoritmo de Cesar para poder encriptar un mensaje de texto en cualquier aplicación de mensajería donde se iniciara con la librería que se estará ocupando para poder garantizar la eficiencia de algoritmo en el programa c, para poder crear este algoritmo se necesito aprender a programar en este lenguaje sencillo se realizará lo siguiente:

- Libros;(Paredes (2006),Kernighan y Ritchie (1991),Grimaldi (1994))

- Articulo revisado ; (Cuzco y cols. (2019)),(Rinza y cols. (2020)),(Fernandez (2004)),(Hellman (1976))

Etapa II. Selección de la herramienta a usar.

En esta investigación se utilizará el lenguaje de programación c, es un lenguaje versátil y poderoso que se utiliza ampliamente en el desarrollo de software y sistemas de computadoras. Donde este programa tiene diversa librería $\#include < Studio - h >$ que es para encabezar estándar de entrada y salida y nos permite manipular fichero desde este lenguaje de programación ,# **include** $< ctype.h >$ esta librería no vas a garantizar diseñar operaciones básica a través de caracteres, $\#include < string.h >$ esta librería contiene un conjunto de funciones para manipular cadena: copiar, cambiar caracteres comparar cadena entre muchas cosas más, ocupando el simulador zinjal para poder compilar el programas en el lenguaje c.

Etapa III. Desarrollo del modelo matemático en el programa.

Se construirá un algoritmo donde nos pueda Encriptar un mensaje de texto aplicando el sistema de cifrado de César donde este algoritmo se encuentra definido matemáticamente de la siguiente forma:

Definición 3.1.1. *Sean* $K = \mathbb{Z}_{27}$*, para* $0 \leq 26$*, se define:*

1. $e_k(x) = x + K \mod 27$ *para el ciframiento.*
2. $e_k(x) = x - K \mod 27$ *para el deciframiento .*

donde x *y* y *perteneces* $\mathbb{Z}_{27}$*.*

También se construirá el algoritmo RSA para Encriptar mensaje de texto donde se encuentra definido matemáticamente de la siguiente forma :

Definición 3.1.2. *Sea* p, q *dos números primos elegidos aleatoriamente, e un número primo relativo con* $(p-1)$ *y* $(q-1)$*, el inverso de e mod* $(p-1)(q-1)$*, y el producto de*

p y q; la codificación de un mensaje con el cifrado RSA se lleva a cabo por medio de la siguiente expresión:

$$C \equiv m^c \ (mod\ n); \tag{3.1}$$

la clase publica de cada (e, n)

$$C \equiv m^d \ (mod\ n); \tag{3.2}$$

donde la clave privada es la letra d

Etapa IV. Evaluación y valoración.

Ya en esta etapa se valoro la eficiencia que tiene ambos algoritmo ya sea algoritmo de César y RSA y comparar cual de esto dos algoritmo para encriptar puede proteger un mensaje de texto y analizar.

CAPÍTULO 4

RESULTADO

Luego de la investigación se realizo el entrenamiento de lo dos algoritmo de criptografía, simultáneamente se efectuaron pruebas para verificar correctamente la funcionalidad de esto sistema.

Los resultados se evaluaron según la protección que le da cada algoritmo a los mensaje de texto y de ahí calificar cual de ello es mas seguro y eficiente para la protección de mensajería. Donde se formo este algoritmo en el lenguaje de programación C para poder observar cada resultado y tenemos lo siguiente código de encriptación:

Algoritmo de César

```
1:  #include <stdio.h>
2:  #include <ctype.h>
3:  #include <string.h>
4:  #include <stdlib.h>
5:  #define LONGITUD_ALFABETO 26
6:  #define INICIO_ALFABETO_MAYUSCULAS 65
7:  #define INICIO_ALFABETO_MINUSCULAS 97
8:  #define MAXIMA_LONGITUD_CADENA 5000
9:  #define MOD(i, n) (i % n + n) % n
10:
11: const char *alfabetoMinusculas = "abcdefghijklmnopqrstuvwxyz",
12: *alfabetoMayusculas = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
13: void cifrar(char *mensaje, char *destino, int rotaciones);
```

```
14: void descifrar(char *mensaje, char *destino, int rotaciones);
15: void GuadarMensaje(char *NameFile,char *MensajeCifrado);
16:
17: int ord(char c);
18:
19: int main(void) {
20:
21:     FILE *Archivo;
22:     int Opcion = 0,Flag = 0,Abortar = 0,Save = 0,
23:  de_donde_decifrar = 0, Cifrado=0;
24:     char NameFile[50];
25:
26: char mensaje[MAXIMA_LONGITUD_CADENA],
27: mensajeCifrado[MAXIMA_LONGITUD_CADENA],
28:         mensajeDescifrado[MAXIMA_LONGITUD_CADENA];
29: int rotaciones;
30:
31:     while(!Flag)
32:     {
33:         do{
34:             printf("Menu: \n");
35:             printf("\t1. cifrar mensaje.\n");
36:             printf("\t2. descifrar mensaje.\n");
37:             scanf("%i",&Opcion);
38:             system("pause");
39:             system("cls");
40:         }while(Opcion < 1 && Opcion > 2);
41:
42:         switch(Opcion)
43:         {
44:         case 1:
```

```
45:
46:             Cifrado =1;
47: printf("Escribe un mensaje para que lo cifre :\n",
48:                    MAXIMA_LONGITUD_CADENA - 1);
49:
50:             fgets(mensaje, MAXIMA_LONGITUD_CADENA, stdin);
51:             mensaje[strcspn(mensaje, "\r\n")] = 0;
52:
53:
54: printf("Escribe el numero de rotaciones :\n");
55:             scanf("%d", &rotaciones);
56:
57:     printf("El mensaje original es: %s\n", mensaje);
58:
59:     cifrar(mensaje, mensajeCifrado, rotaciones);
60:     printf("El mensaje cifrado es: %s\n", mensajeCifrado);
61:
62:     printf(" Desea  Guardar el mensaje en un archivo?\n");
63:     printf(
64:  "Ingrese 1 para guardarlo cualquier otro \n");
65:             scanf("%i",&Save);
66:
67:             if(Save == 1)
68:             {
69:                 do
70:                 {
71:                     if(Abortar == 1)
72:                     {
73:     printf("Ingresa el nuevo nombre del Archivo: \n");
74:                     }else
75:             printf("Ingresa el nombre del Archivo: \n");
```

```
76:
77:                     fflush(stdin);
78:                     gets(NameFile);
79:
80:                     Archivo = fopen(NameFile,"r");
81:                     if (Archivo != NULL)
82:                     {
83: printf("Advertencia el archivo ya existe  Desea  sobre
84: escribirlo?\n");
85:                         printf("Ingrese: 0 (No), 1 (Si)\n");
86:                         scanf("%i",&Abortar);
87:
88:                     }
89:                     if(Abortar == 1 || Archivo == NULL )
90:                     {
91:                         fclose(Archivo);
92:
93:                         GuadarMensaje(NameFile,mensajeCifrado);
94:                         Abortar = 0;
95:
96:                     }
97:
98:                 }while(Abortar == 1);
99:             }
100:
101:             break;
102:         case 2:
103:
104:
105: printf("A continuaci n se desifrara un mensaje: \n");
106: printf("Ingrese: \n1. para descifrar el mensaje actual.\n");
```

```
107: printf("2. Descifrar un mensaje guardado en un Archivo\n");
108:              scanf("%i",&de_donde_decifrar);
109:
110:              switch(de_donde_decifrar)
111:              {
112:              case 1:
113:
114:                  if(Cifrado)
115:                  {
116: descifrar(mensajeCifrado, mensajeDescifrado, rotaciones);
117: printf("El mensaje descifrado es: %s\n", mensajeDescifrado);
118:
119:                  }
120:                  else
121:                  {
122: printf("No hay nada aun para decifrar\n");
123: printf("Menu selecciona la Opcion 1 del menu principal\n");
124:                  }
125:
126:                  break;
127:              case 2:
128:                  printf("ingresa el nombre del Archivo\n");
129:                  fflush(stdin);
130:                  gets(NameFile);
131:
132:                  Archivo = fopen(NameFile,"r+");
133:
134:                  if(Archivo == NULL)
135:                  {
136:                      printf("Archivo no encontrado\n");
137:
```

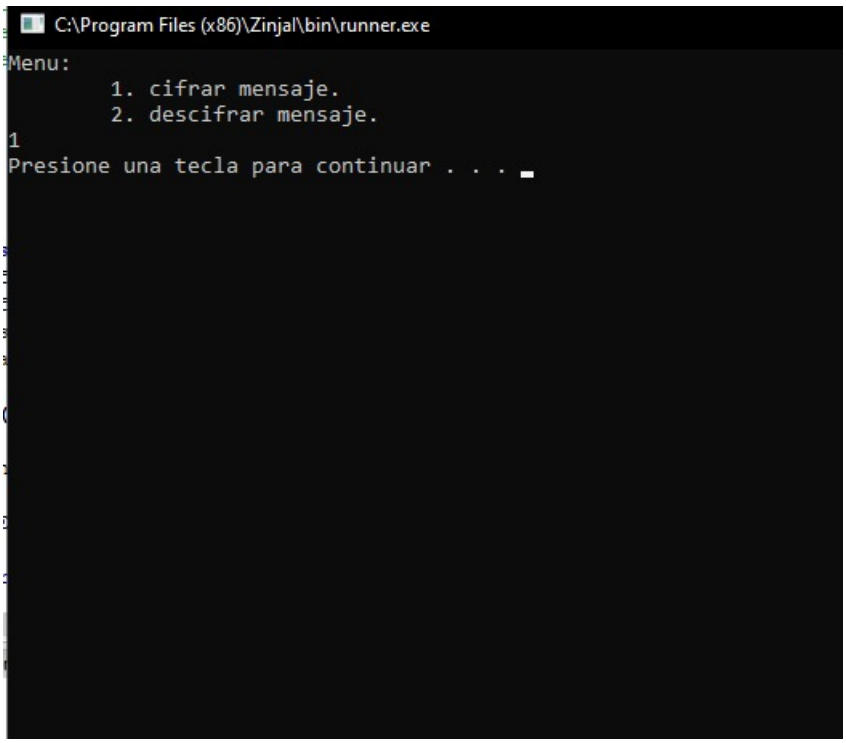
```
138:                     exit(0);
139:                 }
140:
141: printf("Ingresa las rotaciones para desifrar el mensaje: \n");
142:     scanf("%i",&rotaciones);
143:         //fscanf(Archivo,"%s",mensajeCifrado);
144: fgets(mensajeCifrado,MAXIMA_LONGITUD_CADENA,Archivo);
145:
146:     descifrar(mensajeCifrado,mensajeDescifrado,rotaciones);
147:
148:     printf("Mensaje cifrado %s\n",mensajeCifrado);
149:     printf("Extraido del Archivo: %s\n",NameFile);
150:     printf("Mensaje desifrado: %s\n",mensajeDescifrado);
151:                 break;
152:             }
153:
154:
155:             break;
156:         default:
157:             printf("Opcion no valida\n");
158:             break;
159:         }
160:         system("pause");
161:         system("cls");
162:         printf("Desea seguir en el programa?\n");
163:         printf("0 para seguir cualquier otro para no\n");
164:         scanf("%i",&Flag);
165:         system("pause");
166:         system("cls");
167:
168:     }
```

```
169:
170:     return 0;
171: }
172:
173: void GuadarMensaje(char *NameFile,char *MensajeCifrado)
174: {
175:     FILE *Archivo = fopen(NameFile, "w+");
176:
177:     if(Archivo == NULL)
178:     {
179:         perror(NameFile);
180:         exit(0);
181:     }
182:
183:     fprintf(Archivo,"%s",MensajeCifrado);
184:     if(!ferror(Archivo))
185:     {
186:         printf("Se guardo el mensaje con exito\n");
187:     }
188:     fclose(Archivo);
189: }
190:
191:
192:
193: void cifrar(char *mensaje, char *destino, int rotaciones) {
194:
195:     int i = 0;
196:     while (mensaje[i]) {
197:     char caracterActual = mensaje[i];
198:     int posicionOriginal = ord(caracterActual);
199:     if (!isalpha(caracterActual)) {
```

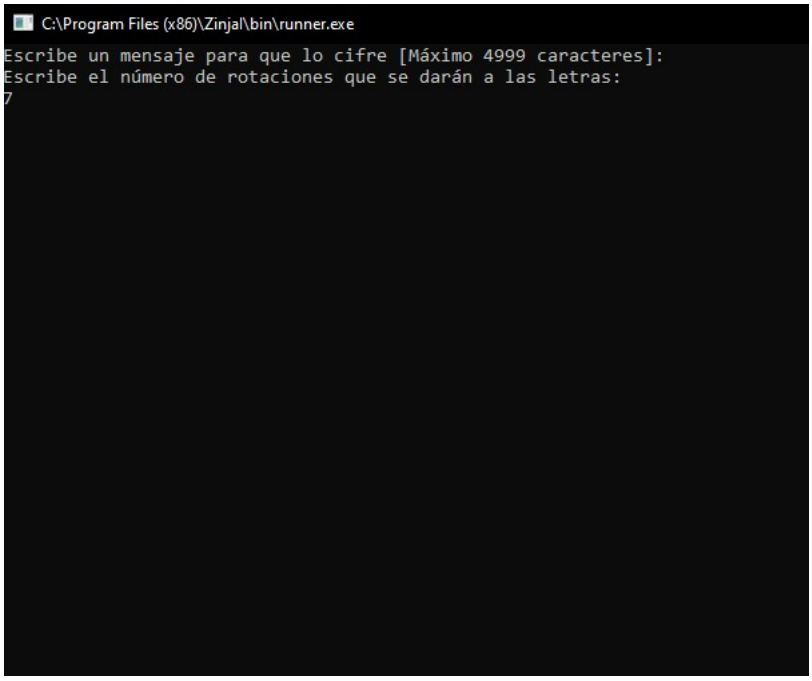
```
200:     destino[i] = caracterActual;
201:     i++;
202:     continue;
203:     }
204:     if (isupper(caracterActual)) {
205:     destino[i] =
206:     alfabetoMayusculas[(posicionOriginal
207:  - INICIO_ALFABETO_MAYUSCULAS +
208:     rotaciones) %
209:     LONGITUD_ALFABETO];
210:     } else {
211:     destino[i] =
212:     alfabetoMinusculas[(posicionOriginal
213:  - INICIO_ALFABETO_MINUSCULAS +
214:     rotaciones) %
215:                 LONGITUD_ALFABETO];
216:         }
217:         i++;
218:     }
219: }
220:
221: void descifrar(char *mensaje, char *destino, int rotaciones) {
222:
223:     int i = 0;
224:     while (mensaje[i]) {
225:         char caracterActual = mensaje[i];
226:         int posicionOriginal = ord(caracterActual);
227:         if (!isalpha(caracterActual)) {
228:             destino[i] = caracterActual;
229:             i++;
230:             continue;
```

```
231:         }
232:         if (isupper(caracterActual)) {
233:             destino[i] = alfabetoMayusculas[MOD(
234: posicionOriginal - INICIO_ALFABETO_MAYUSCULAS - rotaciones,
235:     LONGITUD_ALFABETO)];
236:         } else {
237:             destino[i] = alfabetoMinusculas[MOD(
238: posicionOriginal - INICIO_ALFABETO_MINUSCULAS - rotaciones,
239:     LONGITUD_ALFABETO)];
240:     }
241:     i++;
242:     }
243: }
244: int ord(char c) { return (int)c; }
```

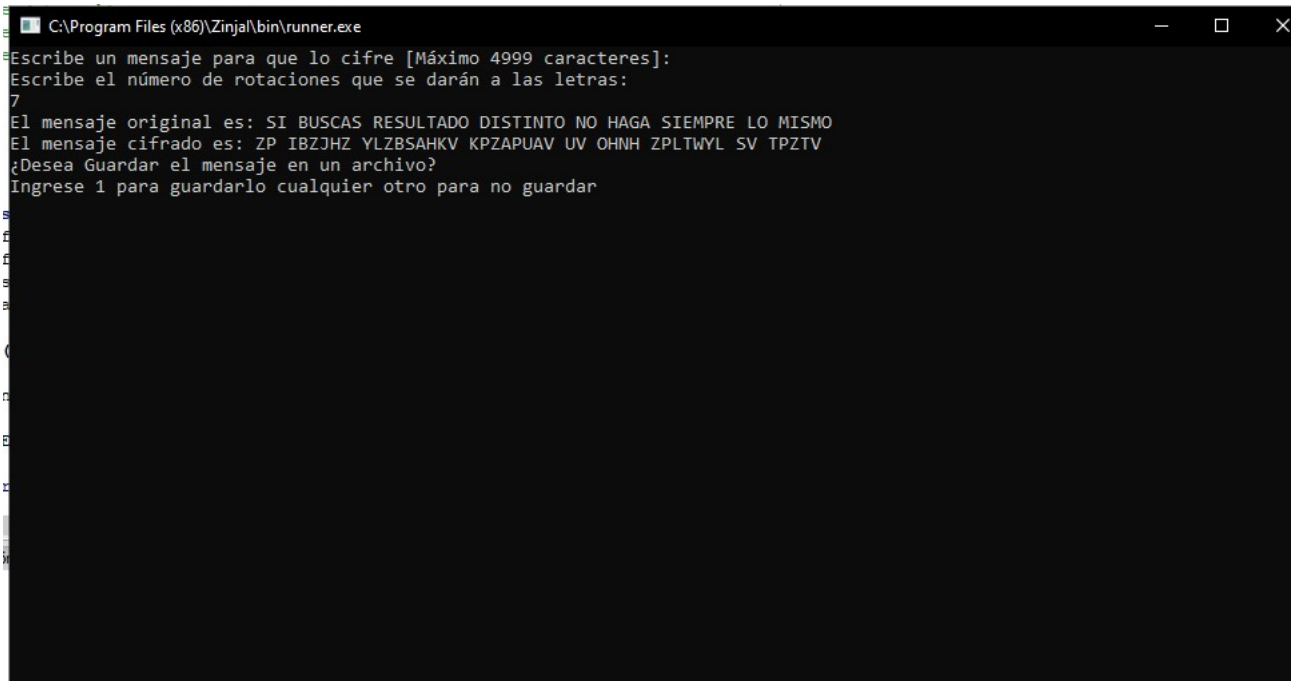
Paso 1: Seleccionar las opciones a ejecutar:



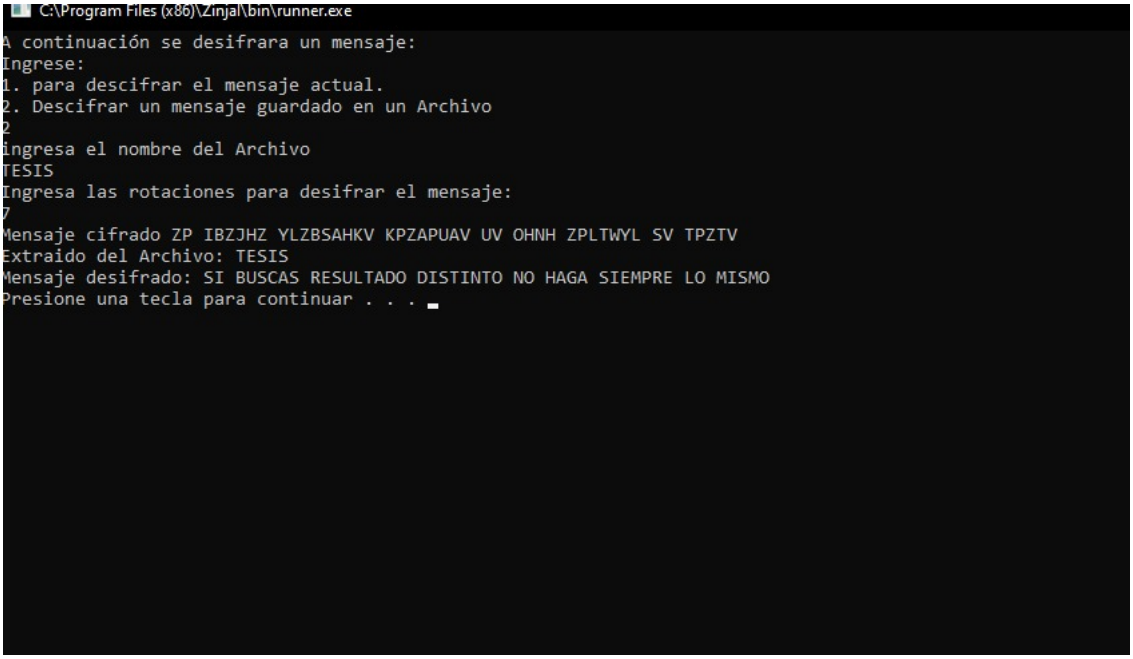
Paso 2: Elegir la rotación del mensaje que se va Encriptar



Paso 3: Mensaje encriptado



Paso 4: Mensaje decifrar el mensaje de texto



Algoritmo RSA

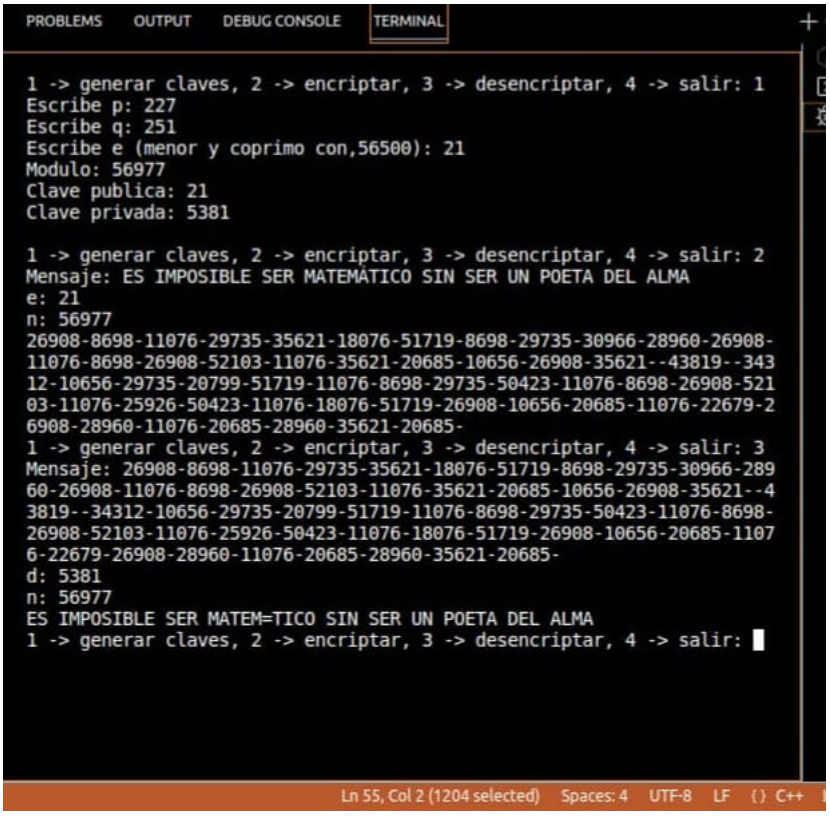
```
1:  #include <iostream>
2:  #include <stdlib.h>
3:  #include <vector>
4:  #include <string>
5:  #include <tuple>
6:  #include "InfInt.h"
7:
8:  using namespace std;
9:
10: InfInt mcd(InfInt a, InfInt b) {
11:     if (b != 0) return mcd(b, a % b);
12:     return a;
13: }
14:
15: // ax + by = mcd(a, b)
16: tuple<InfInt, InfInt, InfInt> euclides(InfInt a, InfInt b) {
```

```
17:     if (a == 0) return make_tuple(b, 0, 1);
18:
19:     InfInt mcd, x, y;
20:     tie(mcd, x, y) = euclides(b % a, a);
21:
22:     return make_tuple(mcd, y - (b / a) * x, x); // (mcd, x, y)
23: }
24: struct RSA {
25:     static void generar();
26:     static vector<InfInt> encriptar(string, InfInt, InfInt);
27:     static string desencriptar(string, InfInt, InfInt);
28: };
29:
30: inline void RSA::generar() {
31:     InfInt p, q, n, phi, e, d;
32:
33:     cout << "Escribe p: ";
34:     cin >> p;
35:
36:     cout << "Escribe q: ";
37:     cin >> q;
38:
39:     n = p * q;
40:     phi = (p - 1) * (q - 1);
41:
42:     do {
43:         cout << "Escribe e (menor y coprimo con,"   << phi << "): ";
44:         cin >> e;
45:     } while (!(e < phi && mcd(e, phi) == 1));
46:
47:     d = get<1>(euclides(e, phi));
```

```
48:     while (d < 0) d += phi;
49:
50:     cout << "Modulo: " << n << endl;
51:     cout << "Clave publica: " << e << endl;
52:     cout << "Clave privada: " << d << endl;
53: }
54:
55: InfInt modpow(InfInt base, InfInt exp, InfInt mod) {
56:     InfInt r = 1;
57:
58:     while (--exp >= 0) {
59:         r *= base;
60:         r %= mod;
61:     }
62:     return r;
63: }
64:
65: inline vector<InfInt> RSA::encriptar(string msg, InfInt e, InfInt n) {
66:     vector<InfInt> resultado;
67:
68:     for (auto c: msg) {
69:         // c^e mod n
70:         resultado.push_back(modpow((int)c, e, n));
71:     }
72:     return resultado;
73: }
74:
75: inline string RSA::desencriptar(string msg, InfInt d, InfInt n) {
76:     vector<InfInt> mensaje;
77:
78:     string temp("");
```

```
79:     for (auto c : msg) {
80:         if (c == '-') {
81:             mensaje.push_back(temp);
82:             temp = "";
83:         } else temp += c;
84:     }
85:     if (temp.length() > 0) mensaje.push_back(temp);
86:
87:     string resultado("");
88:     for (auto c : mensaje) {
89:         // c^d mod n
90:         auto numero = modpow(c, d, n);
91:         resultado += (char)(numero.toInt());
92:     }
93:     return resultado;
94: }
```

El resultado de la incriptarción de mensajes de texto


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1 -> generar claves, 2 -> encriptar, 3 -> desencriptar, 4 -> salir: 1
Escribe p: 227
Escribe q: 251
Escribe e (menor y coprimo con,56500): 21
Modulo: 56977
Clave publica: 21
Clave privada: 5381
1 -> generar claves, 2 -> encriptar, 3 -> desencriptar, 4 -> salir: 2
Mensaje: ES IMPOSIBLE SER MATEMÁTICO SIN SER UN POETA DEL ALMA
e: 21
n: 56977
26908-8698-11076-29735-35621-18076-51719-8698-29735-30966-28960-26908-
11076-8698-26908-52103-11076-35621-20685-10656-26908-35621--43819--343
12-10656-29735-20799-51719-11076-8698-29735-50423-11076-8698-26908-521
03-11076-25926-50423-11076-18076-51719-26908-10656-20685-11076-22679-2
6908-28960-11076-20685-28960-35621-20685-
1 -> generar claves, 2 -> encriptar, 3 -> desencriptar, 4 -> salir: 3
Mensaje: 26908-8698-11076-29735-35621-18076-51719-8698-29735-30966-289
60-26908-11076-8698-26908-52103-11076-35621-20685-10656-26908-35621--4
3819--34312-10656-29735-20799-51719-11076-8698-29735-50423-11076-8698-
26908-52103-11076-25926-50423-11076-18076-51719-26908-10656-20685-1107
6-22679-26908-28960-11076-20685-28960-35621-20685-
d: 5381
n: 56977
ES IMPOSIBLE SER MATEM=TICO SIN SER UN POETA DEL ALMA
1 -> generar claves, 2 -> encriptar, 3 -> desencriptar, 4 -> salir:
Ln 55, Col 2 (1204 selected)
Spaces: 4
UTF-8
LF
{} C++

CAPÍTULO 5

CONCLUSIONES

1. Se obtuvo la aplicación de la congruencia modular en los algoritmos de César y RSA para cifrar texto proporciona una capa adicional de seguridad.Tanto el algoritmo de César como el de RSA se benefician de la congruencia modular al mejorar su resistencia a diversos ataques criptográficos

2. Se comprobó que existe una relación de articulación entre el algoritmo de César, RSA y el lenguaje C, ya que el algoritmo se puede desarrollar y ejecutar de manera satisfactoria.

3. Para Cumplir con la parte práctica, el algoritmo César podría ser utilizado para mensajes que no requieren un alto grado de confidencialidad y donde la simplicidad es prioritaria. Por otro lado, el algoritmo RSA es más adecuado para escenarios en los que la seguridad es una preocupación central, como el cifrado de comunicaciones en línea, transacciones financieras seguras y almacenamiento de datos sensibles.

4. Para concluir, se logró ver la eficiencia que tiene estos dos algoritmos tanto como el algoritmo de César y el algoritmo RSA donde esto sistemas tiene diversa ventaja que permite aplicar la seguridad en la comunicación tipo texto empleando esta dos técnica criptográfica.

CAPÍTULO 6

RECOMENDACIONES

1. El algoritmo Cesar no se debe utilizar en aplicaciones donde se requiere un nivel razonable de seguridad, como la protección de datos sensibles o la comunicación segura.

2. El algoritmo RSA es una opción segura para aplicaciones que requieren niveles razonables de seguridad, como la encriptación de comunicaciones en línea o la firma digital. Sin embargo, debido a su mayor complejidad y tiempo de ejecución, es menos eficiente para cifrar mensajes directamente que los algoritmos de cifrado simétrico.

3. Requiere claves más largas para lograr un nivel razonable de seguridad, lo que puede aumentar los tiempos de operación.

BIBLIOGRAFÍA

Cuzco, R. H., Mantanilla, C. E., Mendez, P. M., y avila, D. F. (2019). Experiencia de aplicación de criptografía para mejorar la seguridad en un método esteganográfico en imágenes. *Revista Espacios*, *40*(38).

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, *31*(4), 469–472.

Fernandez, S. (2004). La criptografía clásica. *Sigma*, *24*(24), 119–141.

Grimaldi, R. P. (1994). *Matemática discreta y combinatoria.* Addison- Wesley Iberoamericana.

Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), 644–654.

Kernighan, B. W., y Ritchie, D. M. (1991). *El lenguaje de programación c.* Pearson Educación.

Lopez, J., y Ruiz, H. (2013). La matemática de la teoría de códigos.

Moreno, G. C. F., y Díaz, E. A. V. (2020). Estudio de la teoría de números aplicada a algunos métodos criptográficos haciendo uso de las tic. *Revista Electrónica de Conocimientos, Saberes y Prácticas*, *3*(1), 49–71.

Paillier, P. (1996). Public key crypto-systems based on composite residue classes. advances in cryptography-eirocrypt'99. *Springer-Verlag LNCS*, *1070*, 72–83.

Paredes, G. G. (2006). Introducción a la criptografía. *Universidad Nacional Autónoma de México. Dirección General de Cómputo*.

Rinza, B. E. S., Sayago, M. P. F., y Rocha-Rinza, T. (2020). Descifrado de cesar por medio de análisis de frecuencia para tres idiomas. *Benemérita Universidad Autónoma de Puebla, UNAM, Facultad de Ciencias de la Computación, Instituto de Química*.

Tominaga García, Y. (2022). Los sistemas de cifrado roce en la seguridad de archivos de texto.

CAPÍTULO 7

ANEXOS

7.1. Introducción a la teoría de código

Cualquier sistema de comunicación consta de cinco partes, a saber, fuente (o emisor), codificador, canal, decodificador y destino (receptor). El canal enlaza un elemento de un conjunto finito de símbolos $A = \{a_1, a_2, ..., a_q\}$ denominado alfabeto del canal. Cada a_i se llama símbolo del canal. Cierta cadena n de símbolos del alfabeto A se llama palabra de longitud n. El conjunto de palabras de longitud n formadas por símbolos de A se denota por A_n y el conjunto de palabras de longitud finita formadas por símbolos de A se denota por A^*. Comúnmente, se representa el alfabeto del canal por un conjunto finito de naturales, de los cuales, el más común es $A = \{0, 1\}$, es decir, el alfabeto binario. El emisor compone los mensajes que se desean enviar a partir de un conjunto finito de símbolos $S = \{s_1, s_2, ..., s_M\}$al cual se le llama alfabeto fuente y cada uno de sus elementos se denomina elemento fuente. Así pues, los mensajes a enviar se consideran palabras de S^*. El codificador transforma o codifica el mensaje a símbolos del canal, mientras que el decodificador realiza la operación contraria, transformando los símbolos del canal a símbolos fuente, con lo que se reconstruye el mensaje original.

Definición 7.1.1. *Un código (bloque) para un alfabeto fuente S y un alfabeto del canal A es una aplicación inyectiva, $C : S \longrightarrow A_n$. La imagen de la aplicación C se denomina conjunto de palabras código, y sus elementos son las palabras código.Lopez y Ruiz (2013)*

7.2. Funcionalidad de cifrado de mensaje alfabético

Paredes (2006) Explica el siguiente ejemplo como manera de transformar un mensaje alfabético en claro (es decir, sin encriptar) en uno numérico, a fin de aplicar el método RSA para encriptarlo. Prescindiremos designos de puntuación y elaboraremos los mensajes con las letras del alfabeto de 26 letras $A, ..., Z$ (la N la ˜podemos sustituir, por ejemplo, por NY), que identificamos con el conjunto de clases $Z_{26} = \{0, 1, .., 25\}$. Así pues, A se identificará con 0, B con 1, etc. Los espacios en blanco convendremos en sustituirlos previamente con una X.

En primer lugar particionaremos el mensaje inicial en mensajes más pequeños de longitud constante k (es decir, de k letras), con el fin encriptar separadamente cada mensaje de longitud k. El valor de k será convenido entre todos los usuarios. Aunque, en aplicaciones reales, es conveniente que el valor de k sea grande.

7.3. Algunos ejemplos RSA y CÉSAR

Fernandez (2004) y Paredes (2006) Explican algunos ejemplos para poder encriptar mensaje de texto lo cuales son:

7.3.1. Ejemplo cifrado de mensaje con el algoritmo de César

Mensaje: “SI BUSCAS RESULTADOS DISTINTOS NO HAGAS SIEMPRE LO MISMO”

1. Se organiza el texto en bloques de 4 letras, para enviarlo y así hacer más difícil la comprensión de este.

SIBU	SCAS	RESU	LTAD	OSDI	STIN
TOSN	OHAG	ASSI	EMPR	ELOM	ISMO

2. Se le asigna a cada letra del mensaje un equivalente numérico, utilizando la tabla 2.

198121	192019	1841921	112003	151938	1920813
20151913	15706	019198	121618	4111512	8191215

3. Se aplica la transformación $e_k(x) \equiv x + K \mod 27$, si $K = 3$ entonces $e_k(x) \equiv x + 3 \mod 27$, donde x corresponde al equivalente numérico de cada letra del mensaje cifrado a enviar, quedando:

$S = 19$

Se reemplaza en

$e_k(x) \equiv x + K \mod 27$

$e_k(19) \equiv 19 + 3 \mod 27$

$e_k(19) \equiv 22 \mod 27$

La primera letra cifrada es $c_1 = 22$, que corresponde a la letra V.

2211424	225322	2172224	142336	1822711	22231116
VLEX	VFDV	UHVX	ÑWDG	RVGL	VWLP

WRVP	RKDJ	DVVL	HOSU	HÑRO	LVOR
23182216	181039	3222211	7151921	7141815	11221518

3. Se aplica la transformación $e_k(x) \equiv x - K \mod 27$, si $K = 3$ entonces $e_k(x) \equiv x - 3 \mod 27$, donde x corresponde al equivalente numérico de cada letra del mensaje descifrado a enviar, quedando:

$V = 22$

Se reemplaza en:

$d_K \equiv y - 3 \mod 27$

$d_{22} \equiv 22 - 3 \mod 27$

$d_{22} \equiv 19 \mod 27$

La primera letra descifrada es d1=19, que corresponde a la letra S

4.Realizando un procedimiento igual al anterior se cifra cada letra del mensaje quedando los bloques de la siguiente manera:

SIBU	SCAS	RESU	LTAD	OSDI	STIN
198121	192019	1841921	112003	151938	1920813

20151913	15706	019198	121618	4111512	8191215
TOSN	OHAG	ASSI	EMPR	ELOM	ISMO

Se organiza la frase para tener el mensaje original: SI BUSCAS RESULTADOS DISTINTOS NO HAGAS SIEMPRE LO MISMO.

7.3.2. Ejemplo Cifrar el siguiente mensaje el cifrado RSA.

Mensaje: "ES IMPOSIBLE SER MATEMÁTICO SIN SER UN POETA DEL ALMA"

1. Se escogen dos números primos p y q, para aplicaciones en software de seguridad se recomienda que los números sean mayores de 200 dígitos no obstante para este caso se van a tomar números primos de solo tres dígitos:

$$p = 227, q = 251$$

Para estos dos números primos se comprobó su primalidad mediante la relación del símbolo de Legendre y Jacobi, para números primos impares de tal manera.

$\frac{a}{b} \equiv a^{\frac{(p-1)}{2}}$ y se cumpla que $r \equiv a^{\frac{n-1}{2}}$ mod$n$ y se compruebe que $r = n - 1$.

2. Se determina el grupo en el que se va a trabajar, es decir el valor $n = pq$

$n = 227 \cdot 251$ $n = 56977$

se obtiene que se va a trabajar en el grupo $\mathbb{Z}_{56977}$

3. Se determinar el orden de grupo $\phi(n) = (p-1)(q-1)$

$\phi(n) = (227-1)(251-1)$

$\phi(n) = 226 \cdot 250$

$\phi(n) = 56500$

4. Se escoge el valor de e de tal manera que el m.c.d$(\phi(n), e) = 1$ Se selecciona e= 21 y se comprueba que aplicando el algoritmo de la división: m.c.d$(\phi(56500), 21) = 1$ y, además

$56500 = 2690 \cdot 21 + 10$

$21 = 2 \cdot 10 + 1$

Es decir, $e = 21$ es comprimo con $\phi(n) = 56500$. Por lo tanto, la clave pública es $(e, n = (21, 56977)$.

$N =$ Tamaño del alfabetos (27 letras)

$(j-1, j)$ Tamaño del bloque de entrada y salida.

$$N^{j-1} < n < N^j$$

$$27^3 < 56977 < 27^4$$

Los bloques que se van a formar para cifrar el mensaje son de 3 cifras.

6. Se expresa e=21 como suma de potencias de 2, esto se puede conseguir a partir de la expresión binaria de un número.

$21 = 101012 = 16 + 4 + 1$

7.Se expresa el mensaje a su respectivo equivalente numérico.

51900	91316	15199	2125	00195	180013	1205	13120	9315
ESoo	IMP	OSI	BLE	ooSE	RooM	ATE	MAT	ICO

ooSI	NooS	ERoo	UNoo	POE	TA00	DEL	ooAL	MA
00199	180019	51800	211400	16155	20100	4512	00112	1301

El número cero representa el espacio entre las palabras del mensaje a cifrar, por lo que el equivalente numérico se corre una unidad.

8. Utilizando la ecuación se reemplaza para cifrar cada uno de los bloques.

$c_1 = ESoo = 51900$

Se expresa en base 27 para que el mensaje sea un elemento del grupo $\mathbb{Z}_{56977}$.

$ES = 5 \times 27^2 + 19 \times 27 + 0 \times 27^0 = 4158(mod\ 56977)$

Quedando $c_1 = 4158^{21}$ mod 56977,ahora se encriptar c_1 con la clave pública (21,56977),aplicando la exponencial rápida el proceso de cifrado se realiza mediante: $a^{2i} = (a^{2i-1})^2$

$$
\begin{aligned}
C_1 &= 4158^{16+4+1}(mod\ 56977) \\
&= 4158^{16} = 4158^{24} = (4158^{23}) \cdot (4158^2 3) \cdot (\ mod\ 56977) \\
&= (4158^{22})^2 \cdot (4158^{22})^2 \cdot (\ mod\ 56977) \\
&= ((4158)^{21})^2)^2 \cdot ((4158)^{21})^2)^2 \cdot (\ mod\ 56977) \\
&= ((4158))^2)^2) \cdot ((4158)^2)^2) \cdot (\ mod\ 56977) \\
&= ((4158)^2) \cdot (4158)^2 (\ mod\ 56977) \\
&= 42952 \cdot 42952 (mod\ 56977) \\
&= 16021 (\ mod\ 56977)
\end{aligned}
$$

Ahora se trabaja con

$$
\begin{aligned}
4158^4 = 4158^{22} &= (4158^{21}) \cdot (4158^{21}) (\ mod\ 56977) \\
&= 24933 \cdot 24933 (\ mod\ 56977) = 35419 (\ mod\ 56977)
\end{aligned}
$$

$$4158 = 4158 = 4158 (\ mod\ 56977)$$

por lo tanto

$c_1 = 4158^{16+4+1} (\ mod\ 56977) = 16021 \cdot 35419 (\ mod\ 56977) = 9501 (\ \text{mód}\ \ 56977)$

Se decodifica c_1 a base 27, para saber a qué letras equivale.

$c_1 = 9501 \longrightarrow 13024$

El primer bloque para enviar es: 13 0 24, donde el número 13 equivale a la letra M, el cero es el espacio y el número 24 se reemplaza por la letra X; es decir, el mensaje codificado es M X. Realizando el procedimiento del punto 8 se codifica los otros bloques de letras que pertenecen al mensaje:

MXC,UOB,YDB,KHA, RKT,BMI,JQB,CZU,ABS,BGH,MJE,LBG,TDBQ,

GYX,WZU,CID,ZUB KIH,NRS.