UNIVERSIDAD DE PANAMÁ

FACULTAD DE CIENCIAS NATURALES EXACTAS Y TECNOLOGÍA

ESCUELA DE MATEMÁTICA

FUNDAMENTOS MATEMÁTICOS DE LA SEGURIDAD DIGITAL: UN ANÁLISIS DEL ALGORITMO RSA EN LA CRIPTOGRAFÍA MODERNA

TESIS PARA OPTAR POR EL TÍTULO DE

LICENCIADO EN MATEMÁTICA

ESTUDIANTE

VICTOR VALENZUELA

Ced.: 4-706-1043

PANAMÁ

2026

DEDICATORIA

Primero, agradezco a Dios por permitirme alcanzar este importante logro en mi vida académica. Dedico este trabajo con profundo amor y respeto a la memoria de mi padre, Oliver Valenzuela, que Dios lo tenga en su gloria. Sé que, dondequiera que esté, se siente orgulloso de mí.

Su ejemplo de dedicación y esfuerzo diario fue mi mayor inspiración para seguir adelante, enfrentar cada desafío con valentía y no rendirme ante las dificultades. Papá, este logro es para ti. Gracias por enseñarme, con tu ejemplo, el valor del trabajo duro y la perseverancia.

AGRADECIMIENTO

En primer lugar, le agradezco a Dios por darme la salud, la sabiduría y la fortaleza necesarias para poder culminar esta tesis; sin Él, nada de esto habría sido posible. A mis padres, quienes me inculcaron el valor de la perseverancia y la importancia de no rendirme nunca ante los desafíos, les debo gran parte de este logro.

A mi esposa, Mirla, por ser mi compañera incondicional en cada paso del camino. Su apoyo constante, aliento y amor han sido fundamentales en este proceso, tanto en los momentos de alegría como en los de dificultad. Extiendo también mi gratitud a mi suegra, Antonia, quien con sus consejos y respaldo ha sido como una segunda madre para mí.

A toda mi familia, que siempre estuvo presente con palabras de aliento y apoyo incondicional, les agradezco profundamente por creer en mí y acompañarme en este recorrido.

Finalmente, quiero expresar mi sincero agradecimiento a mi asesor, el Profesor Pedro Saucedo, por su guía, paciencia y valiosos consejos, los cuales fueron esenciales para la realización y finalización de este trabajo académico.

RESUMEN

En esta tesis examinaremos los fundamentos tanto teóricos y prácticos de la criptografía, enfocándonos particularmente en el algoritmo RSA, uno de los métodos de cifrado asimétrico más utilizados en la actualidad. En el primer capítulo hablamos un poco de la historia de la criptografía, desde sus orígenes en la antigüedad utilizando métodos simples de codificación, hasta llegar a los métodos más complejos utilizados en nuestra época actual. Todos estos avances en la criptografía van evolucionando en paralelo con el avance de la matemática.

En el capítulo 2, nos enfocamos en los conceptos matemáticos que son fundamentales para el desarrollo de la criptografía, entre los que podemos mencionar: la teoría de números, aritmética modular y la algebra abstracta. Estos conceptos son explicados para proporcionar al estudiante de matemática, una comprensión de las herramientas que hacen posible la criptografía moderna.

El capítulo 3 esta dedico a la implementación práctica del algoritmo RSA, paso a paso analizamos el algoritmo mediante un script diseñado para fines educativos, se harán todos los procesos matemáticos manualmente y se compararán los valores obtenidos con la salida del algoritmo RSA.

PALABRAS CLAVES

Criptografía, algoritmo RSA, conceptos matemáticos, implementación practica

ABSTRACT

In this work, we examine both the theoretical and practical foundations of cryptography, focusing particularly on the RSA algorithm, one of the most widely used asymmetric encryption methods today. In the first chapter, we discuss the history of cryptography, from its origins in antiquity using simple encoding methods to the more complex techniques employed in the modern era. All these advances in cryptography have evolved in parallel with the development of mathematics.

In Chapter 2, we focus on the mathematical concepts that are fundamental to the development of cryptography, including number theory, modular arithmetic, and abstract algebra. These concepts are explained to provide mathematics students with an understanding of the tools that make modern cryptography possible.

Chapter 3 is dedicated to the practical implementation of the RSA algorithm. We analyze the algorithm step by step through a script designed for educational purposes, performing all mathematical processes manually and comparing the obtained values with the RSA algorithm output.

KEYWORDS

Cryptography, RSA algorithm, mathematical concepts, practical implementation

ÍNDICE GENERAL

INTRODUCCIÓN

La criptografía ha sido, a lo largo de la historia, una disciplina fundamental para la protección de la información. Desde los primeros métodos de cifrado utilizados en la antigüedad hasta los algoritmos criptográficos modernos, su evolución ha estado estrechamente vinculada al desarrollo de la matemática y la tecnología. En la actualidad, la criptografía desempeña un papel esencial en ámbitos como las telecomunicaciones, la banca, la ciberseguridad y el comercio electrónico, donde la protección de los datos es una prioridad.

Dentro de la criptografía moderna, uno de los algoritmos más relevantes es el algoritmo RSA, un sistema de cifrado asimétrico ampliamente utilizado para garantizar la seguridad en la transmisión de información. Su fortaleza se basa en conceptos matemáticos como la teoría de números y la aritmética modular, lo que lo convierte en un esquema robusto y eficiente.

El presente trabajo tiene como objetivo analizar los fundamentos teóricos y matemáticos de la criptografía, con un enfoque particular en la implementación del algoritmo RSA. La investigación busca proporcionar a los estudiantes de la Licenciatura en Matemática una base sólida que les permita comprender la estructura matemática sobre la cual se construyen los sistemas criptográficos modernos, así como la aplicación práctica de dichos conceptos en la protección de la información.

Para el desarrollo de esta investigación, la tesis se estructura en tres capítulos:

- **Capítulo 1:** Se presenta una visión general de la criptografía, abordando su historia desde la antigüedad hasta la era moderna. Se explican los métodos de cifrado más relevantes que han sido utilizados a lo largo del tiempo, así como su relación con el desarrollo de las matemáticas.
- **Capítulo 2:** Se detallan los conceptos matemáticos fundamentales para la criptografía, incluyendo la teoría de números, la aritmética modular y el álgebra abstracta. Estos elementos proporcionan el marco teórico necesario para comprender los principios sobre los que se construyen los sistemas criptográficos actuales.

- **Capítulo 3:** Se centra en la implementación del algoritmo RSA, explicando paso a paso su funcionamiento y su aplicación práctica. Se analiza un script desarrollado para fines educativos, donde se detallan los cálculos matemáticos realizados manualmente y se comparan con los resultados obtenidos a través del algoritmo.

Se presentan las conclusiones y recomendaciones, resaltando la importancia del estudio de la criptografía en la formación matemática y su aplicación en la seguridad de la información. Además, se proponen líneas de investigación futura para aquellos estudiantes que deseen profundizar en la criptografía y explorar nuevas áreas de desarrollo en esta disciplina.

El propósito de esta investigación es proporcionar a los estudiantes de Licenciatura en Matemáticas una base sólida y accesible en criptografía, permitiéndoles visualizar la conexión entre las matemáticas y la seguridad informática. A través del estudio del algoritmo RSA, los estudiantes podrán comprender cómo se aplican conceptos matemáticos en la protección de la información, facilitando así su incursión en estudios más avanzados dentro de la criptografía y la ciberseguridad.

CAPÍTULO 1

INTRODUCCIÓN A LA CRIPTOGRAFÍA

LA CRIPTOGRAFÍA

1.1 Definiciones Fundamentales

1.1.1 ¿Qué es la criptografía?

Desde un enfoque técnico, la criptografía se centra en el diseño y análisis de algoritmos matemáticos que permiten proteger la información frente a accesos no autorizados.

La palabra criptografía tiene su origen etimológico en el griego, derivado de los términos **κρυπτός (kryptós)**, que significa "oculto" o "secreto", y **γραφή (graphḗ)**, que significa "escritura". Así, etimológicamente, la criptografía puede definirse como la **"escritura oculta"** o **"escritura secreta"**.

Desde el punto de vista matemático, la criptografía se considera una rama de la teoría de la información, estrechamente relacionada con la teoría de códigos y la compresión de datos. Estas disciplinas comparten el objetivo de estudiar la transmisión de información a través de un canal, donde siempre intervienen dos agentes principales: un emisor y un receptor. La criptografía, en particular, se centra en garantizar que la comunicación sea segura y que solo el receptor autorizado pueda interpretar correctamente el mensaje.

La criptografía se fundamenta en la matemática, que proporciona las bases teóricas necesarias para el diseño y análisis de sistemas criptográficos. Se apoya principalmente en áreas como la teoría de números (números primos y aritmética modular), el álgebra abstracta (estructuras como grupos y cuerpos), y la teoría de la complejidad computacional, que evalúa la dificultad de resolver ciertos problemas matemáticos. Estas herramientas permiten garantizar la seguridad, integridad y autenticidad de la información, haciendo de la matemática un pilar esencial en el desarrollo de la criptografía moderna, tanto en el ámbito teórico como en sus aplicaciones prácticas.

1.1.2 Texto Claro y Texto Cifrado

El **texto claro**, también conocido como **texto plano**, es la representación original y legible de la información que se desea proteger. Este texto no ha sido modificado ni transformado por un algoritmo de cifrado, por lo que puede ser comprendido fácilmente por cualquier persona que lo lea.

Ejemplo de texto claro:

> *"La contraseña para acceder al sistema es 1234."*

En este caso, cualquier receptor que reciba este mensaje puede interpretarlo directamente.

El **texto cifrado** es la representación transformada o codificada del texto claro tras haber sido procesado por un algoritmo de cifrado utilizando una clave. El texto cifrado es ilegible e incomprensible sin la clave necesaria para descifrarlo, lo que garantiza la confidencialidad de la información.

Ejemplo de texto cifrado: Si el texto claro es:

> *"La contraseña para acceder al sistema es 1234."*

Después de aplicarse un algoritmo de cifrado con una clave, podría transformarse en:

"K3@9x!PQ&5zL"

Este texto es incomprensible para cualquiera que no posea la clave correcta para descifrarlo.

Relación entre Texto Claro y Texto Cifrado

El proceso de transformación de texto claro a texto cifrado (y viceversa) depende de dos elementos fundamentales:

1. **El algoritmo criptográfico**: Es el conjunto de reglas que define cómo se realiza el cifrado y descifrado.
2. **La clave criptográfica**: Es el valor único que determina la manera en que el algoritmo procesa la información.

Importancia en la Criptografía

Desempeñan un papel fundamental en la seguridad de la información, garantizando la confidencialidad, integridad y autenticidad de los datos. La confidencialidad se logra al transformar el texto claro en texto cifrado, asegurando que solo las personas autorizadas puedan acceder a la información. Por otro lado, la integridad protege el contenido contra modificaciones

no autorizadas durante su transmisión, evitando alteraciones que puedan comprometer la veracidad del mensaje. Finalmente, la autenticidad permite verificar que el mensaje proviene de una fuente confiable al ser descifrado, asegurando la identidad del remitente. Estos principios son esenciales para el funcionamiento de cualquier sistema criptográfico, ya que su propósito principal es transformar la información en una forma segura, protegiéndola durante su transmisión o almacenamiento frente a accesos no autorizados.

1.1.3 Claves criptográficas

Las claves criptográficas son elementos fundamentales en los sistemas de criptografía, ya que son los valores que permiten realizar las operaciones de cifrado y descifrado de la información. En términos simples, una clave es una cadena de datos utilizada para transformar un mensaje original (texto claro) en un mensaje cifrado (texto cifrado) y viceversa.

Definición de Clave Criptográfica

Una clave criptográfica es un conjunto de datos, típicamente una secuencia de números o caracteres, que se utiliza en combinación con un algoritmo criptográfico para cifrar o descifrar información. Su secreto y cualidad de única, irrepetible y singular son esenciales para garantizar la seguridad del sistema criptográfico.

En términos matemáticos, la clave actúa como un parámetro que controla las operaciones del algoritmo criptográfico, garantizando que solo las partes autorizadas puedan acceder a la información protegida.

Tipos de Claves Criptográficas

Las claves criptográficas pueden clasificarse en dos grandes categorías, dependiendo del tipo de algoritmo con el que se utilicen:

Claves Simétricas

En este tipo de sistema, se utiliza una única clave para realizar tanto el cifrado como el descifrado.

- **Características:**
    - ✓ Rápidas y eficientes en términos computacionales.

- ✓ Requieren que la clave se comparta de manera segura entre las partes.

- **Ejemplo Matemático:**
  - ✓ En un cifrado como el AES (Advanced Encryption Standard), se utiliza una clave simétrica de 128, 192 o 256 bits.
- **Ventajas:**
  - ✓ Eficiencia computacional.
  - ✓ Adecuadas para cifrar grandes volúmenes de datos.
- **Desventajas:**
  - ✓ La clave debe transmitirse de forma segura, lo que puede ser un desafío.

Claves Asimétricas

En este sistema se utilizan dos claves distintas pero relacionadas matemáticamente:

1. **Clave pública**: Se utiliza para cifrar el mensaje y puede ser compartida libremente.
2. **Clave privada**: Se utiliza para descifrar el mensaje y debe mantenerse secreta.

- **Características**:
  - ✓ Las claves están basadas en problemas matemáticos complejos, como la factorización de números primos o el logaritmo discreto, como es el caso del algoritmo RSA.
- **Ventajas**:
  - ✓ No es necesario compartir la clave privada, lo que mejora la seguridad.
- **Desventajas**:
  - ✓ Más lentas que los sistemas simétricos.

- **Ejemplo Matemático**:
    - ✓ En RSA, el cifrado utiliza la clave pública (e, n) y el descifrado la clave privada (d, n), cuyas fórmulas se desarrollarán en el capítulo 3.

$$C = M^e \mod n \quad \text{(cifrado)}$$

$$M = C^d \mod n \quad \text{(descifrado)}$$

Propiedades de las Claves Criptográficas

Para garantizar la seguridad de un sistema criptográfico, las claves deben cumplir con ciertas propiedades fundamentales:

1. **Longitud adecuada**:
    - Claves más largas proporcionan mayor seguridad, ya que aumentan la dificultad de romper el cifrado mediante fuerza bruta.
    - Ejemplo:
        - ✓ Claves simétricas: 128, 192 o 256 bits.
        - ✓ Claves asimétricas: Mínimo 2048 bits para RSA.
2. **Aleatoriedad**:
    - Las claves deben ser generadas de forma aleatoria para evitar patrones predecibles que puedan comprometer la seguridad.
3. **Secreto**:
    - En sistemas simétricos, la clave debe mantenerse confidencial para ambas partes.
    - En sistemas asimétricos, la clave privada debe mantenerse completamente segura.

4. **Unicidad**:

    - Cada clave debe ser única para evitar colisiones y garantizar la integridad de los datos cifrados.

Importancia Matemática de las Claves Criptográficas

Las claves criptográficas tienen una importancia matemática fundamental dentro de la criptografía, ya que constituyen el elemento esencial para garantizar la seguridad en los sistemas de cifrado. Su fortaleza radica en principios matemáticos como la teoría de números, la aritmética modular y los problemas computacionalmente difíciles, que impiden que un atacante pueda descifrar la información sin poseer la clave correspondiente.

En el caso de los algoritmos de cifrado asimétrico, como RSA, la seguridad se basa en la dificultad de factorizar números enteros muy grandes, lo que hace que el cálculo de la clave privada a partir de la clave pública sea prácticamente imposible en tiempos razonables con la tecnología actual. Por otro lado, en los algoritmos simétricos, la fortaleza radica en la generación de claves aleatorias y en funciones matemáticas que garantizan la resistencia frente a ataques de fuerza bruta.

1.1.4 Algoritmos criptográficos

Los algoritmos criptográficos constituyen un conjunto de pasos matemáticos y computacionales que permiten realizar operaciones de cifrado y descifrado sobre los datos. Es un procedimiento bien definido que transforma los datos en una forma segura mediante el uso de una clave. Este procedimiento garantiza que solo quienes tengan la clave puedan recuperar la información original. Son la base de la criptografía y están diseñados para proteger la confidencialidad, integridad y autenticidad de la información en un entorno digital.

Fórmula general:

- **Cifrado**: $C = E_k(P)$

    - ✓ C: Texto cifrado.
    - ✓ P: Texto claro.
    - ✓ E_k: Algoritmo de cifrado que usa la clave k.

- **Descifrado**: $P=D_k(C)$

    - ✓ D_k: Algoritmo de descifrado que usa la clave k.

Clasificación de los Algoritmos Criptográficos

Los algoritmos criptográficos se clasifican en tres categorías principales según su propósito y la forma en que utilizan las claves:

1. **Algoritmos de Cifrado Simétrico**
    - Utilizan una única clave para cifrar y descifrar datos.
    - **Ejemplo:**
        - ✓ Algoritmo AES (Advanced Encryption Standard).
    - **Funcionamiento**:
        - ✓ La misma clave k se usa para transformar el texto claro (P) en texto cifrado (C) y viceversa.
    - **Ventajas**:
        - ✓ Son rápidos y eficientes.
    - **Desventajas**:
        - ✓ La clave debe ser compartida de manera segura entre las partes.
2. **Algoritmos de Cifrado Asimétrico**
    - Usan un par de claves:
        - ✓ **Clave pública**: Para cifrar.
        - ✓ **Clave privada**: Para descifrar
    - **Ejemplo**:
        - ✓ Algoritmo RSA.
    - **Funcionamiento**:
        - ✓ La clave pública se comparte abiertamente, pero solo el poseedor de la clave privada puede descifrar el mensaje.

- **Ventajas**:
    - ✓ No es necesario compartir la clave privada.
- **Desventajas:**
    - ✓ Son más lentos que los algoritmos simétricos.

3. **Algoritmos de Hash**
    - Transforman un mensaje de cualquier longitud en un resumen de longitud fija.
    - **Ejemplo**:
        - ✓ SHA-256.
    - **Funcionamiento:**
        - ✓ Generan un valor único para los datos de entrada, lo que permite verificar su integridad.
    - **Ventajas**:
        - ✓ No requieren claves.
    - **Desventajas**:
        - ✓ No permiten descifrar el mensaje original.

Ejemplos de Algoritmos Criptográficos

1. **Algoritmos Simétricos**:
    - **DES (Data Encryption Standard)**: Utiliza claves de 56 bits.
    - **AES (Advanced Encryption Standard)**: Más seguro, con claves de 128, 192 o 256 bits.
2. **Algoritmos Asimétricos**:
    - **RSA**: Basado en la factorización de números grandes.
    - **Diffie-Hellman**: Permite el intercambio de claves de manera segura.
    - **ECC (Cryptografía de Curvas Elípticas)**: Usa curvas algebraicas para generar claves más seguras con tamaños más pequeños.

3. **Funciones Hash**:

    - **SHA-256**: Genera un hash de 256 bits.
    - **MD5**: Antiguo, pero menos seguro debido a vulnerabilidades conocidas.

Importancia Matemática en los Algoritmos Criptográficos

Los algoritmos criptográficos se basan en principios matemáticos fundamentales:

- **Aritmética Modular**: Utilizada en RSA y Diffie-Hellman.
- **Teoría de Números**: Importante en sistemas asimétricos como RSA.
- **Álgebra Abstracta**: Base de la criptografía de curvas elípticas.
- **Complejidad Computacional**: Garantiza que ciertos problemas sean computacionalmente inviables de resolver sin la clave correcta.

1.1.5 Cifrados Polialfabéticos

Los cifrados polialfabéticos son un método de cifrado por sustitución en el cual se utilizan múltiples alfabetos en lugar de uno solo. Esto significa que una misma letra del texto claro puede ser cifrada de diferentes maneras dependiendo de su posición en el mensaje y de la clave utilizada. Este enfoque incrementa la seguridad frente a ataques de análisis de frecuencias, que es una de las técnicas comunes que se utiliza para romper cifrados monoalfabéticos como el cifrado César.

Historia

El cifrado polialfabético fue introducido en el siglo XV por el matemático y criptógrafo italiano León Battista Alberti, quien es considerado el padre de la criptografía moderna debido a sus innovaciones en el campo del cifrado. Alberti desarrolló un método en el que se utilizaban múltiples alfabetos de sustitución para cifrar un mensaje, lo que hacía que los intentos de criptoanálisis fueran mucho más difíciles en comparación con los cifrados monoalfabéticos, como el Cifrado César.

Su invención representó un avance significativo en la seguridad de la información, ya que introdujo la idea de cambiar el alfabeto de sustitución en distintas partes del mensaje, dificultando la identificación de patrones en el texto cifrado. Además, Alberti diseñó un disco

de cifrado, conocido como el Disco de Alberti, el cual permitía cambiar dinámicamente el alfabeto cifrado en función de una clave.

El desarrollo del cifrado polialfabético marcó un punto clave en la evolución de la criptografía, ya que permitió la creación de sistemas más sofisticados que sentaron las bases para los cifrados modernos utilizados en la actualidad.

Funcionamiento Matemático

El cifrado polialfabético se basa en la transformación de letras del texto claro (P) a texto cifrado (C) utilizando una clave (K).

Fórmulas generales:

1. **Cifrado**:

   Como explicación utilizaremos la fórmula que proviene del cifrado de Vigenére, un método clásico de criptografía por sustitución polialfabética

   $$C_i = (P_i + K_i) \bmod 26$$

Donde :

- P_i: Letra en la posición i del texto claro.
- K_i: Letra correspondiente de la clave.
- C_i: Letra cifrada en la posición i.
- mod 26: se usa porque hay 26 letras en el alfabeto latino, el módulo asegura que el resultado sea siempre un número dentro del rango 0 a 25 (correspondiente a una letra válida)

2. **Descifrado**:

   $$P_i = (C_i - K_i) \bmod 26$$

1.2 Origen y Evolución de la Criptografía

Las Raíces de la Criptografía que es el arte de escribir y descifrar códigos, se remontan a muchos años atrás en la historia. Desde los inicios de la civilización, las personas han tratado de buscar formas de transmitir los mensajes de manera segura, asegurándose que solo el destinatario previsto lograra entender el significado.

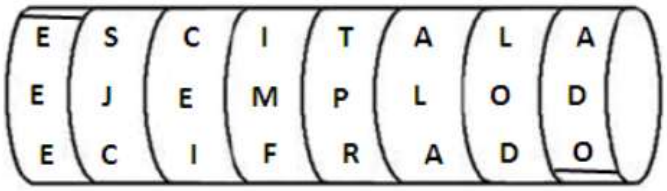
Los primeros métodos de criptografía eran simples pero ingeniosos, Se empleaban en situaciones críticas relacionadas con la seguridad personal, gubernamental y militar.

1.2.1 La Escítala Espartana

Uno de los primeros métodos documentados de criptografía fue la escítala espartano que se remonta al siglo V a.C., aproximadamente hace 2.500 años atrás, en la antigua Grecia, los espartanos usaban un sistema secreto de escritura durante las guerras con Atenas. Este sistema de codificación secreto en realidad era algo simple como explicamos a continuación; tanto el emisor como el receptor del mensaje debían poseer un cilindro de madera (la escítala), con bases de igual radio, el que enviaba el mensaje enrollaba una tira de cuero o pergamino como si se tratara de una venda, y una vez enrollada escribía el mensaje en forma longitudinal, de la siguiente manera (texto en claro: "**ESCITALAEJEMPLODECIFRADO"**):

Figura 1

La escítala



(Fuente: Autor desconocido, tomado de Revista Digital Universitaria, UNAM, 2006)

De esta manera si algún enemigo interceptaba el mensaje no podría entender el mensaje, ya que leería en la cinta desenrollada el mensaje cifrado:

"EEESJCCEIIMFTPRALALODADO".

Las únicas personas capaces de descifrar el mensaje son las que tendría las dimensiones del cilindro correcto, y al enrollar nuevamente la tira en el cilindro leerían el mensaje codificado

1.2.2 El Cifrado César

En el siglo I a. C., el legendario general romano Julio César crea un método de cifrado que emplea por primera vez una transformación al texto claro de tipo monoalfabética. Este cifrado simple aplicaba un desplazamiento constante de tres caracteres al texto claro, en otras palabras,

consistía en asignar a cada letra del alfabeto la letra que estaba tres lugares más a su derecha, tomando el criterio lógico que la letra Z se empezaba de nuevo por la letra A. De este modo el alfabeto cifrado es el mismo que el alfabeto del texto claro, pero desplazado tres espacios hacia la derecha módulo n, con n como el número de letras de este.

En la imagen siguiente se muestra el alfabeto y la trasformación que realiza este cifrador por sustitución de caracteres al alfabeto castellano de 27 letras, vemos que las letras fueron desplazadas con la letra correspondiente tres espacios hacia la derecha.

Figura 2

Cifrado de César

Alfabeto	A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
Alfabeto	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

(Fuente: Autor desconocido, tomado de Revista Digital Universitaria, UNAM, 2006)

Realicemos un ejemplo como seria el cifrado de un mensaje:

Mensaje Original: **MENSAJE DE PRUEBA**

Mensaje Cifrado: **OHPVDMN GH SUXHED**

Como podemos darnos cuenta la clave del cifrado de César está en el número “3”, este estilo de cifrado monoalfabéticos de sustitución perduraron a lo largo del tiempo, por ejemplo, la Orden de Los Templarios también usaba en el siglo XII un método criptográfico de sustitución que asociaba a cada letra del alfabeto un símbolo gráfico.

1.2.3 El cifrado por transposición

Durante la edad media, se desarrollaron diferentes métodos como el cifrado por transposición, el cual consiste en tratar de colocar un fragmento de texto cifrado en un lugar conocido previamente por el destinatario del mensaje. Este cifrado podría ser: alternando el orden natural de las letras, silabas o palabras, a veces también formando anagramas con ellas.

La diferencia de este cifrado con el de cifrado de sustitución donde las letras son reemplazadas por otras, en el de transposición las letras del mensaje original permanecen iguales, pero su orden se altera siguiendo un patrón específico.

Uno de los cifrados de transposición más conocido de esa época es el cifrado de la columna o cifrado columnar.

Ejemplo:

Mensaje original (Texto Claro)

"**DEFENDER EL CASTILLO**"

Se eliminan los espacios entre letras del mensaje y se acomoda en una tabla por columnas, supongamos que este caso la palabra clave es "**MESA**" (tiene 4 letras, entonces se crea una tabla con 4 columnas).

M	**E**	**S**	**A**
D	E	F	E
N	D	E	R
E	L	C	A
S	T	I	L
L	O	X	X

Se escribió el mensaje en forma corrida por fila, y se rellenaron los espacios vacíos al final con X, esto se hace si la cantidad de leras no llenan todas las celdas al terminar el mensaje.

Se ordenan las columnas ahora en el orden alfabético de la clave, la palabra **MESA** que es la palabra clave tiene un orden alfabético como sigue: A (1era posición), E (2da posición), M (3era posición) y S (4ta posición).

Al aplicar ese orden, las columnas se presentan de la siguiente manera:

A	**E**	**M**	**S**
E	E	D	F
R	D	N	E
A	L	E	C
L	T	S	I
X	O	L	X

Se lee las letras columna por columna según el orden establecido por la clave.

1. **A → E R A L X**
2. **E → E D L T O**
3. **M → D N E S L**
4. **S → F E C I X**

El mensaje cifrado seria "**ERALX EDLTO DNESL FECIX**" colocando las letras de las columnas en orden.

Eliminando los espacios entre las letras el mensaje cifrado final sería:

"**ERALXEDLTODNESLFECIX"**

METODO PARA DESCIFRAR EL MENSAJE

Para descifrar el mensaje debemos hacer el proceso contrario, ya que sabemos que la palabra clave es "**MESA**", armamos la tabla con las columnas de las letras en orden alfabético "**AEMS**", como esta en la tabla anterior y colocamos las letras por columna en orden, nos debe quedar la misma tabla anterior.

Ya que tenemos esa tabla colocamos la palabra clave en orden "**MESA**", y colocamos el mensaje en las filas de 4 columna siguiendo ese orden. Nos debe quedar igual a la tabla inicial. Y se lee fila por fila para recuperar el mensaje original

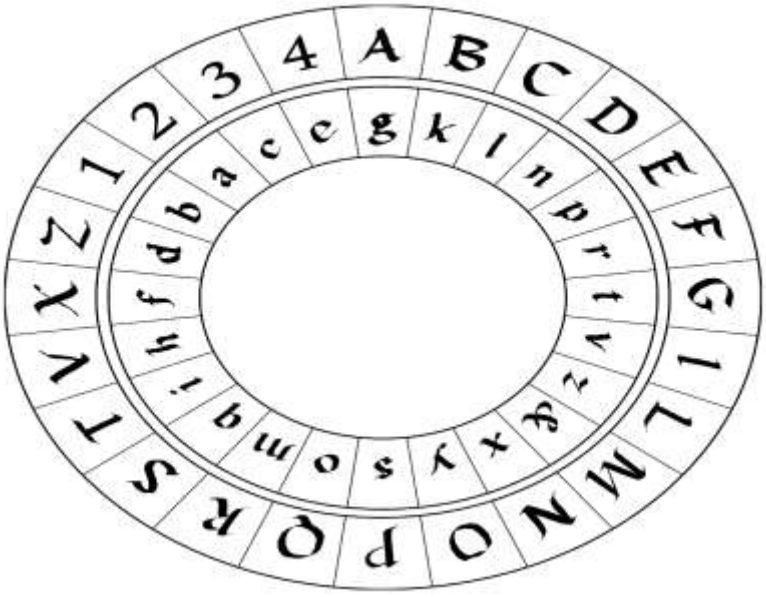
1.2.4 El disco de Alberti

León Battista Alberti, criptógrafo italiano nacido en Génova, Italia es conocido como el padre de la criptografía moderna. Para el año de 1466 realiza su gran aportación a la criptografía, el disco de Alberti, el cual se considera el primer sistema polialfabético de la historia. Un sistema polialfabético significa que un mismo carácter se sustituye cada vez por otro carácter distinto siguiendo un cierto criterio de codificación.

El disco de Alberti presenta en su círculo exterior los 20 caracteres del alfabeto latino utilizados en ese sistema, esto es, los mismos del alfabeto castellano excepto las letras H, J, Ñ, K, U, W e Y, y se incluyen los números 1, 2, 3 y 4 para códigos especiales. Por su parte, en el disco interior aparecen todos los caracteres del latín además del signo & y las letras H, K e Y. Al ser 24 los caracteres representados en cada disco, es posible definir hasta 24 sustituciones diferentes; es

Figura 3

El disco de Alberti



decir, dependiendo de la posición del disco interior la cantidad máxima de alfabetos de cifrado es igual a 24.

Para cifrar un mensaje, una vez establecida la correspondencia entre caracteres de ambos discos o, lo que es lo mismo, el alfabeto de cifrado, se repasa letra a letra el texto en claro del disco exterior y se sustituye cada una de ellas por la letra correspondiente del disco interior.

Para nuestro ejemplo supongamos que alineamos los discos como muestra la figura 3, En este caso, la clave inicial es **A → g**, es decir, la letra "A" del disco externo se alinea con la "g" del disco interno.

Supongamos que queremos cifrar la palabra **"SECRETO"** con la clave inicial **A → g**.

Texto claro	S	E	C	R	E	T	O
Texto Cifrado	Q	p	L	M	P	i	y

El mensaje cifrado seria: "**qplmpiy**"

Para descifrar el mensaje se invierte el proceso, buscamos ahora en las letras del disco interno y recuperamos la correspondiente del disco externo, recordar que los discos están alineados con la indicación **A → g.**

q → S, p → E, l → C, m → R, p → E, i → T, y → O

recuperando el mensaje original: "**SECRETO**"

1.2.5 El cifrado de Vigenère

Este método criptográfico es una variante del cifrado de César, fue creado por el criptólogo francés Blaise de Vigenére en el siglo XVI, el método es un cifrado polialfabético donde utilizamos una llave secreta para realizar la codificación del mensaje. Este método fue considerado prácticamente indescifrable hasta el siglo XIX, cuando descubrieron técnicas para romperlo.

Como hemos mencionado el cifrado de Vigenére es una variación del cifrado de César, de la misma forma que esta, toma las letras del alfabeto desde 0 a 25 para el idioma inglés y desde 0 a 26 en el caso del alfabeto español tradicional, considerando la inclusión de la letra Ñ.

Figura 4

Tabla del alfabeto con su numeración correspondiente

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Fuente: (Cinvestav, s.f.)

Lo primero que se debe hacer para cifrar con este método es elegir una llave que suele ser una o varias palabras, en nuestro ejemplo, utilizaremos la llave "**OSCAR**"

Si vemos en la tabla de la figura 4, su representación numérica es K= (**14,18,2,0,17**)

Figura 5

Mensaje cifrado

Texto Claro	e	s	t	o		e	s		u	n	a		p	r	u	e	b	a
Llave	o	s	c	a		r	o		s	c	a		r	o	s	c	a	r
Texto Cifrado	s	k	v	o		v	g		m	p	a		g	f	m	g	b	r

Fuente: (Cinvestav, s.f.)

El mensaje cifrado sería: "**skvo vg mpa gfmgbr**"

Este mensaje cifrado se obtuvo sumando el valor de la letra del texto claro (mensaje original), sumándole el valor de la letra correspondiente de la clave que está en módulo 26, estos valores lo podemos ver en la tabla de la figura 4, de esta manera el valor numérico de la letra del texto claro "e" = 4 y el valor de la letra correspondiente de la llave es "o" = 14. Sumando estos dos valores nos da como resultado 18, y el valor numérico 18 en la tabla de la figura 4, corresponde a la letra "**s**" y así de la misma forma hacemos con las demás letras para obtener el mensaje cifrado.

Para hacer el descifrado del mensaje, debemos restar el valor de la llave a el texto cifrado, de la forma siguiente, el valor del texto cifrado de la letra "s" = 18, y el valor de la letra de la llave correspondiente es "o"= 14, al restar s – o, 18 -14 = 4, el valor número 4 en la tabla representa a la letra "e", de esta manera hacemos con las demás letras para recuperar el mensaje original.

Existe una tabla de Vigenére, la cual permite hacer el cifrado más fácilmente evitando trabajar con números, esta tabla la podremos ver en los anexos de esta tesis, con su explicación de su uso, si el lector está interesado en conocer más sobre este método de codificación.

1.2.6 Incorporación de la aritmética modular

Con la obra *Disquisitiones Arithmeticae* de Carl Friedrich Gauss, la aritmética modular se formalizó y se convirtió en la base de muchos sistemas criptográficos modernos. La aritmética modular es un concepto fundamental en la teoría de números y como hemos mencionado se utiliza ampliamente en el campo de la ciberseguridad, particularmente en la criptografía clásica, se puede decir que forma la columna vertebral de varios algoritmos y protocolos criptográficos.

La aritmética modular implica la aritmética de restos. Cuando un número entero ***a*** se divide por otro número entero ***n*** (conocido como módulo), produce un cociente ***q*** y un resto ***r.*** Se expresa mediante la ecuación siguiente: $a = nq + r$

donde $0 \leq r < n$. El resto ***r*** es el resultado de la operación modular, y esto se escribe como:

$$r = a \bmod n$$

El concepto de congruencia es central en la aritmética modular. Dos números enteros ***a*** y ***b*** se dice que son congruente módulo ***n*** si al dividirlos tienen el mismo resto ***n***. Esto se escribe como:

$$a \equiv b \bmod n$$

La aritmética modular se utilizó en varios métodos de cifrados históricos como es el caso del cifrado César y el cifrado Vigenére, esto cifrados se pueden representar de manera matemática de la siguiente forma $C = (P + k) \bmod 26$. Este concepto matemático sentó las bases para la criptografía moderna basada en números.

Componentes:

1. **C**: Representa el carácter **cifrado**, es decir, la letra resultante después de aplicar el cifrado.
2. **P**: Es el carácter del **texto plano** (mensaje original), convertido a su valor numérico en el alfabeto:
    - Ejemplo: A=0, B=1, C=2,...,Z=25.
3. **k**: Es la **clave** del cifrado, que indica el número de posiciones que se desplazará cada letra en el alfabeto.
4. **mod 26:** Es la operación **módulo**, que asegura que el resultado siempre se mantenga dentro del rango de 0 a 25, ya que el alfabeto tiene 26 letras.

El descifrado se realiza de mediante la siguiente forma:

$$P = (C - k) \bmod 26$$

Aplicaciones en criptografía moderna

La aritmética modular no es sólo una reliquia de los cifrados históricos, sino también una piedra angular de los algoritmos criptográficos modernos. Algunos de los protocolos criptográficos más utilizados, como RSA (Rivest-Shamir-Adleman) y el intercambio de claves Diffie-Hellman, dependen en gran medida de la aritmética modular.

1.2.7 Teoría de números

Podemos decir que la teoría de números empezó con el matemático griego Diofanto de Alejandría en el siglo III d.C. Diofanto escribió trece libros (siete de los cuales se han perdido) dedicados a la resolución de ecuaciones algebraicas, intentando dar métodos para encontrar sus soluciones enteras o racionales.

Pero la contribución (indirecta) más importante de Diofanto fue a partir de la traducción al latín de los seis primeros libros con el nombre de Aritmética en 1621 por C.G. Bachet. Esta traducción fue la que inspiró al verdadero padre de la teoría de números, ***Pierre de Fermat.***

El desarrollo de la teoría de números por matemáticos como Pierre de Fermat, Leonhard Euler y Carl Friedrich Gauss proporcionó herramientas fundamentales para la criptografía. Conceptos como los números primos, la función totiente de Euler y el pequeño teorema de Fermat permitieron la creación de algoritmos de clave pública, como el sistema RSA, que garantiza la seguridad mediante la factorización de números grandes.

1.2.8 Álgebra abstracta

El álgebra abstracta se desarrolló notablemente entre los siglos XIX y XX, rompió con los cánones clásicos y se centró en un enfoque más abstracto de los conceptos fundamentales de la aritmética y los sistemas algebraicos. En el siglo XX, se consolidó como "álgebra moderna", aplicándose en diversas áreas de la matemática. Los trabajos de Gauss generalizaron numerosas estructuras algebraicas.

Históricamente el desarrollo matemático de la criptografía se puede situar alrededor del año 1948 cuando Shannon establece las bases matemáticas de la teoría de la información al publicar *Communication Theory of Secrecy Systems* en donde expone un algoritmo cifrado irrompible.

El álgebra abstracta y la criptografía se han desarrollado en interrelación a lo largo de los siglos XIX y XX. En el ámbito del álgebra abstracta, la teoría de grupos, que estudia estructuras

algebraicas como los grupos, ha sido fundamental en diversos campos, incluida la criptografía. Desde el siglo XIX, con contribuciones como las de F. W. Kasiski, se comenzaron a establecer métodos para romper cifrados, lo que llevó a un mayor interés en las estructuras algebraicas que subyacen a la creación y el rompimiento de códigos. Así, el álgebra moderna ha influido significativamente en la forma en que se entiende y se implementa la criptografía en las matemáticas y la computación.

En la criptografía moderna, el Álgebra Abstracta juega un papel fundamental en el diseño y análisis de algoritmos criptográficos seguros. Mediante el uso de estructuras algebraicas abstractas, como grupos y campos finitos, permite desarrollar algoritmos criptográficos eficientes y resistentes a los ataques.

Por ejemplo, en el cifrado de clave pública, se utilizan conceptos del Álgebra Abstracta, como los grupos cíclicos y los problemas matemáticos difíciles, para garantizar la seguridad del intercambio de claves. Estos algoritmos, como el RSA y el Diffie-Hellman, se basan en la dificultad de resolver ciertos problemas matemáticos, como el factorizar números grandes, para proteger la información.

Además, el Álgebra Abstracta también se utiliza en la construcción de funciones de hash criptográficas, que son fundamentales en la integridad de los datos y la autenticación. Estas funciones, como el SHA-256, se basan en propiedades de los grupos y los anillos para garantizar que cualquier modificación en los datos sea detectada.

Un ejemplo concreto de cómo se aplica el Álgebra Abstracta en la criptografía es el algoritmo de curva elíptica, que utiliza conceptos de geometría algebraica para asegurar la confidencialidad y la integridad de los datos. Este algoritmo se basa en la idea de trabajar con puntos en una curva elíptica y utilizar operaciones especiales, como la suma y la multiplicación, para realizar operaciones criptográficas.

1.2.9 La máquina Enigma

Una de las herramientas criptográficas más famosas de la historia es la máquina Enigma, creada a principios del siglo XX, este dispositivo electromecánico de cifrado fue utilizado por el ejército alemán durante la segunda guerra mundial.

Utilizaba un sistema de rotores para cifrar mensajes, basando su funcionamiento en el cilindro de Jefferson, creado por el presidente de Estados Unidos Thomas Jefferson en el siglo XVII. Su diseño consistía en un teclado parecido a una máquina de escribir, un conjunto de rotores giratorios que cambiaban constantemente el cifrado y un tablero de conexiones que permitía modificar la correspondencia entre las letras, añadiendo más complejidad a la codificación.

Al momento de presionar una tecla, una señal eléctrica pasaba por los rotores y el tablero de conexiones, generando una letra cifrada diferente. Adicional a esto después de cada pulsación, los rotores giraban, cambiando la configuración del cifrado en cada letra escrita, haciendo del cifrado extremadamente complejo y difícil de romper.

A pesar de toda esta sofisticación la maquina Enigma fue descifrada gracias a los esfuerzos de equipo de matemáticos y criptógrafos aliados. Alan Turing desarrollo la famosa "Bomba Criptológica", una máquina que era capaz de descifrar los mensajes de Enigma a gran velocidad. Este logro acorto la guerra en al menos dos años y salvo millones de vidas.

Figura 6

Máquina Enigma

(Fuente: Autor desconocido, tomado de Revista Digital Universitaria, UNAM, 2006)

Para la Criptografía, la tecnología desarrollada con estas máquinas sentó las bases para la criptografía moderna, inspirando el uso de cifrados basados en transformaciones matemáticas complejas, algunos de sus principios de sustitución y permutación se siguen empleando en algoritmos de cifrado actuales como el AES y RSA.

1.2.10 Teoría de la complejidad computacional

La teoría de la complejidad computacional estudia las necesidades de memoria, tiempo y otros recursos computacionales para resolver problemas; de esta manera es posible explicar por qué unos problemas son más difíciles de resolver que otros. La teoría de la complejidad computacional actúa como un guía en el territorio de problemas informáticos, así como un mapa que nos indica los caminos hacia soluciones más eficientes. Sin embargo, nos enfrentamos al desafío de entender por qué algunas rutas son más directas que otras.

En el ámbito de la criptografía, esta teoría se convierte en una herramienta esencial utilizada para evaluar la fortaleza de los sistemas criptográficos existentes y para fomentar la innovación en el desarrollo de nuevas técnicas de codificación. Su propósito fundamental radica en garantizar la confidencialidad y seguridad de la información en el mundo de la criptografía.

La teoría de la complejidad computacional tiene sus orígenes en los matemáticos como Alan Turing y Claude Shannon, quienes comenzaron a analizar la seguridad de los sistemas criptográficos en función del tiempo y los recursos necesarios para resolver problemas matemáticos. Ejemplos clave incluyen la factorización de números grandes, utilizada en RSA, y el problema del logaritmo discreto, base de algoritmos como Diffie-Hellman y ECC.

1.2.11 Funciones Hash y teoría de probabilidad

Una función criptográfica hash- usualmente conocida como “hash”- es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija. Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.

En el contexto de la teoría de probabilidades, la probabilidad de colisión (donde diferentes entradas producen la misma salida hash) es un aspecto crítico; esta probabilidad puede aumentar más rápidamente de lo que se espera, lo que se debe considerar al diseñar funciones hash seguras.

Las funciones hash, como SHA-256, introdujeron nuevas herramientas para garantizar la integridad de los datos y verificar la autenticidad de la información. Estas herramientas son esenciales en aplicaciones modernas como firmas digitales y blockchain.

Es importante recordar que el estudio de estas funciones y su comportamiento probabilístico ha sido fundamental en áreas como la criptografía y el almacenamiento de datos a partir del siglo XX hasta la actualidad.

1.2.12 Criptografía poscuántica

La criptografía poscuántica es el desarrollo de nuevos tipos de enfoques criptográficos que se puedan implementar en los ordenadores convencionales de hoy pero que a la vez resulten inmunes a los ataques de los ordenadores cuánticos de mañana.

Las tecnologías emergentes están listas para traer cambios transformadores al campo de la criptografía, con la computación cuántica destacando como un desarrollo particularmente significativo. La computación cuántica aprovecha los principios de la mecánica cuántica para procesar información de manera fundamentalmente diferente en comparación con las computadoras clásicas. Los bits cuánticos, o qúbits, pueden existir simultáneamente en múltiples estados, lo que permite a las computadoras cuánticas realizar cálculos complejos a velocidades sin precedentes.

Una de las principales implicaciones de la computación cuántica para la criptografía es su potencial para romper sistemas criptográficos ampliamente utilizados. Muchos de los métodos de encriptación que actualmente protegen las comunicaciones digitales globales, como RSA y ECC, se basan en la dificultad computacional de ciertos problemas matemáticos, como factorizar grandes enteros o resolver logaritmos discretos. Algoritmos cuánticos, especialmente el algoritmo de Shor, podrían resolver estos problemas de manera eficiente, volviendo vulnerables a las técnicas criptográficas actuales.

Como resultado, el campo está explorando activamente estrategias para desarrollar técnicas de encriptación resistentes a la computación cuántica. Esto ha dado lugar a la floreciente área de investigación de la criptografía postcuántica, que busca crear algoritmos que puedan resistir los ataques de computadoras cuánticas.

CAPÍTULO 2

CONCEPTOS MATEMÁTICOS CLAVES EN CRIPTOGRAFÍA

CONCEPTOS MATEMÁTICOS CLAVES EN CRIPTOGRAFÍA

La criptografía, como disciplina científica, tiene sus fundamentos en las matemáticas; se podría decir que, desde sus inicios, esta ciencia ha necesitado de herramientas matemáticas para diseñar y analizar métodos para proteger la información. Actualmente la criptografía moderna se basa en conceptos matemáticos avanzados con lo que garantiza la seguridad de sistemas, en estos tiempos donde el mundo está cada vez más interconectado.

El desarrollo de algoritmos criptográficos como RSA y AES no serían posibles sin el uso de la aritmética modular, teoría de números y álgebra abstracta, entre otros conceptos matemáticos claves. Estas herramientas permiten generar claves seguras, realizar cifrados complejos y sobre todo garantizar la integridad de los datos, enfrentándose a desafíos en el futuro como la computación cuántica y los ataques de fuerza bruta.

En este capítulo veremos los conceptos matemáticos que forman la base de la criptografía moderna. Se estudiará como la aritmética modular facilita los cálculos cíclicos, como la teoría de números es fundamental en algoritmos como RSA, y como el álgebra abstracta forma parte del diseño de sistemas como AES.

Lo que buscamos, es mostrar el vínculo que existe entre las matemáticas y la criptografía, destacando cómo esta relación ha permitido el desarrollo de sistemas robustos y seguros, los cuales se siguen utilizando diariamente.

2.1 Aritmética Modular

La aritmética modular fue introducida en 1801 por Carl Friedrich Gauss en su libro Disquisitiones Arithmeticae. Algunas veces se le llama, aritmética del reloj, ya que los números “dan la vuelta” tras alcanzar cierto valor llamado módulo. La Aritmética Modular juega un rol clave dentro de la Teoría de Números, Criptografía y el Álgebra Abstracta.

Actualmente, en la era de la información, ha cobrado especial importancia debido, entre otros factores, a la necesidad de las grandes empresas de mantener mucha información de forma segura y de acceso fácil y rápido; siendo precisamente a través del uso de códigos, en los que se aplica la factorización de números grandes, que se cubre esta necesidad.

2.1.1 Definición

La aritmética modular se define matemáticamente como:

$$a \mod n = r$$

donde:

- a: Es el número inicial
- n: Es el módulo
- r: Es el residuo de la división de a entre n

Por ejemplo:

$$17 \mod 5 = 2 \qquad \text{ya que} \quad 17 = 3 \times 5 + 2$$

2.1.2 Propiedades Claves

Una vez definida la aritmética modular, es posible analizar algunas de sus propiedades fundamentales, las cuales resultan esenciales en criptografía.

1. **Suma modular:**

$$(a + b) \mod n = [(a \mod n) + (b \mod n)] \mod n$$

Ejemplo:

Si a = 14, b = 9, y n = 5

$$(14 + 9) \mod 5 = [(14 \mod 5) + (9 \mod 5)] \mod 5$$

$$23 \mod 5 = (4 + 4) \mod 5$$

$$3 = 8 \mod 5$$

$$3 = 3$$

2. **Multiplicación modular:**

$$(a \times b) \mod n = [(a \mod n) \times (b \mod n)] \mod n$$

Ejemplo:

Si $a = 7$, $b = 3$, y $n = 4$

$$(7 \times 3) \mod 4 = [(7 \mod 4) \times (3 \mod 4)] \mod 4$$

$$21 \mod 4 = (3 \times 3) \mod 4$$

$$1 = 9 \bmod 4$$

$$1 = 1$$

3. **Exponenciación modular:**

$$(a^b) \bmod n$$

Esta operación es central en algoritmos como RSA.

Ejemplo:

Si $a = 3$ y $b = 4$, y $n = 5$

$$(3^4) \bmod 5$$

$$81 \bmod 5 = 1$$

4. **Inverso modular**

Para los enteros a y n, el inverso modular de a es un número x tal que:

$$a \times x \equiv 1 \bmod n$$

Ejemplo:

Si $a = 3$ y $n = 7$, el inverso modular de 3 es el 5

Para comprobar esto probamos valores diferentes para x:

$x = 1 \Rightarrow 3 \times 1 = 3 \bmod 7 \neq 1$
$x = 2 \Rightarrow 3 \times 2 = 6 \bmod 7 \neq 1$
$x = 3 \Rightarrow 3 \times 3 = 9 \equiv 2 \bmod 7 \neq 1$
$x = 4 \Rightarrow 3 \times 4 = 12 \equiv 5 \bmod 7 \neq 1$
$x = 5 \Rightarrow 3 \times 5 = 15 \equiv 1 \bmod 7 = 1$

Hemos logrado encontrar que $x = 5$ cumple con la ecuación

Respuesta: El inverso modular de 3 módulo 7 es 5, debido a que:

$3 \times 5 = 15 \equiv 1 \bmod 7$

Para números mucho más grandes, se utiliza el algoritmo extendido de Euclides, el cual se explicará con detalles en este mismo capitulo.

2.1.3 Aplicación en Criptografía

1. **Cifrado de César:**

Este cifrado utiliza una aritmética modular para realizar el desplazamiento cíclico sobre el alfabeto. La fórmula matemática es la siguiente:

$$\mathbf{C = (P + k) \mod 26}$$

Donde:

- C: Letra cifrada.
- P: Letra original (texto claro).
- K: Cantidad de desplazamientos o clave.

Ejemplo:

Si deseamos codificar la letra del alfabeto "A" con una cantidad de desplazamiento o clave de 3 posiciones, tendremos:

P = A (0), el valor numérico de A en el alfabeto es de (0) y el valor de K = 3 (tres posiciones)

C = (0 + 3) mod 26 esto da como resultado a 3, C = 3, y el valor numérico 3 en el alfabeto corresponde a la letra D. Al cifrar la letra A nos da la letra D.

$$\mathbf{C = D}$$

2. **RSA (Rivest-Shamir-Adleman):**

Se utiliza la aritmética modular en este algoritmo para realizar las operaciones de cifrado y descifrado con claves públicas y privadas. La fórmula matemática es la siguiente:

Para el Cifrado: $\mathbf{C = M^e \mod n}$

Para el descifrado: $\mathbf{M = C^d \mod n}$

Donde:

- M: Mensaje en texto claro.
- C: Texto cifrado.
- e, d: Exponente de la clave pública y privada, respectivamente.
- n: Producto de dos números primos grandes.

El siguiente ejemplo se presenta únicamente con fines ilustrativos y educativos; en la práctica, los valores de p y q deben ser números primos grandes.

Lo primero que debemos hacer es seleccionar dos números primos:

$$p = 2 \text{ y } q = 11$$

Con estos dos números primos se calcula el valor de n

$$n = p \times q = 22$$

El siguiente paso es calcular la función totiente de Euler **ϕ(n)**

$$\phi(n) = (p - 1)(q - 1) = (2 - 1)(11 - 1) = 1 \times 10 = 10$$

Ahora debemos elegir un valor para e, que debe tener las siguientes condiciones; Debe ser un número positivo, $1 < e < \phi(n)$ y debe ser coprimo con **ϕ(n)**.

MCD (e, ϕ(*n*)) = 1, con esto se garantiza que exista un inverso modular *d*, necesario para el cifrado. Vamos a elegir **e = 3**.

Debemos calcular ahora el valor para ***d***, que debe ser el inverso modular de e **mod ϕ(*n*).**

$$3d \equiv 1 \bmod 10$$

Realizamos el procedimiento que ya hemos explicado para calcular el inverso modular y nos da ***d* = 7** porque:

$$\mathbf{3 \times 7 = 21 \bmod 10 = 1}$$

Ya tenemos las claves generadas tanto pública como la privada:

- **Clave pública:** $(e, n) = (3,22)$
- **Clave privada:** $(d, n) = (7,22)$

Supongamos que queremos cifrar el mensaje m = 5, aplicamos la fórmula:

$$C \equiv m^e \bmod n$$

$$C \equiv 5^3 \bmod 22 = 125 \bmod 22$$

$$C = 15$$

Texto cifrado: C = 15

Para descifrarlo realizamos el procedimiento siguiente:

$$m \equiv C^d \bmod n$$

$$m \equiv 15^7 \bmod 22$$

Calculamos el valor de $15^7 \bmod 22$:

1. $15^2 \equiv 5 \bmod 22$
2. $15^4 \equiv 3 \bmod 22$
3. $15^7 \equiv 5 \bmod 22$

Dándonos como resultado el valor para **m = 5** para $15^7 \bmod 22$, y con esto recuperamos el mensaje original.

También la aritmética modular está presente en los algoritmos de Diffie-Hellman que permiten el intercambio seguro de claves mediante cálculos modulares basados en potencias y en el cifrado basado en las curvas elípticas que realizan operaciones definidas sobre puntos, que dependen de la aritmética modular.

2.1.4 Ventajas de la Aritmética Modular en Criptografía

La aritmética modular constituye la base matemática de numerosos algoritmos criptográficos modernos. Su uso permite trabajar de manera estructurada con números grandes y establecer relaciones de congruencia fundamentales para los procesos de cifrado y descifrado en sistemas como RSA y AES.

A partir de estos conceptos, se hace necesario introducir la teoría de números, ya que proporciona las herramientas matemáticas esenciales para el estudio de números primos, factorización y congruencias utilizadas en criptografía.

2.2 Teoría de Números

2.2.1 Definición

La teoría de números es la rama de las matemáticas que estudia el concepto de número, sus definiciones, tipos, relaciones y aplicaciones. En el ámbito de la criptografía, esta disciplina proporciona herramientas fundamentales para el diseño de algoritmos de cifrado, ya que trabaja con conceptos como los números primos, la factorización, y las congruencias.

Los sistemas de criptografía asimétrica, como el algoritmo RSA, se basan directamente en problemas relacionados con la teoría de números, especialmente la dificultad de factorizar números grandes y calcular raíces modulares.

2.2.2 Conceptos Claves de la Teoría de Números en Criptografía

Los conceptos claves de la Teoría de Números en Criptografía incluyen a los números primos, también destacan el Teorema de Euler y la función totient, utilizados en la generación de claves y en operaciones modulares avanzadas, y el Teorema Pequeño de Fermat, aplicado en pruebas de primalidad y en la optimización de cálculos modulares. Estos conceptos forman la base matemática de los algoritmos criptográficos modernos y garantizan su seguridad y eficiencia.

1. Números Primos:

Definición:

Un número primo es un número entero mayor que 1 que solo tiene dos divisores: él mismo y el 1. Es decir, son números que no pueden descomponerse en factores menores de manera exacta (es decir, los números compuestos). A esta condición se le conoce como primalidad.

Importancia en la Criptografía:

Los números primos son fundamentales en criptografía, ya que proporcionan la base matemática para la seguridad de muchos algoritmos de cifrado. Su principal importancia radica en la dificultad de factorizar números grandes, lo que fortalece sistemas como RSA al hacer prácticamente imposible obtener la clave privada a partir de la clave pública. Además, se utilizan en la generación de claves seguras, garantizando que los datos cifrados sean resistentes a ataques. Sus propiedades matemáticas únicas permiten su aplicación en algoritmos modulares como Diffie-Hellman y ElGamal, donde aseguran operaciones eficientes y seguras. También juegan un papel clave en la resistencia a ataques computacionales, ya que el uso de primos grandes dificulta la factorización y protege contra posibles ataques criptográficos futuros. En resumen, los números primos son esenciales para la seguridad de la información en la era digital.

Una propiedad importante que podemos mencionar de los números primos, según el teorema de Euclides, es que estos son infinitos.

2. El Pequeño Teorema de Fermat

El Pequeño Teorema de Fermat fue formulado por el matemático Pierre de Fermat (1607-1665), considerado uno de los precursores de la teoría de números. Fermat trabajaba como abogado y

matemático aficionado, pero sus contribuciones fueron fundamentales en la matemática moderna.

A lo largo de su vida, Fermat escribió muchas de sus conjeturas en los márgenes de libros y en cartas a otros matemáticos, sin proporcionar demostraciones formales. En una carta enviada a su amigo Bernard Frénicle de Bessy en 1640, Fermat afirmó el resultado que hoy conocemos como su Pequeño Teorema.

Definición:

"Si p es un número primo y a es un número entero no divisible por p, entonces $a^{p-1} \equiv 1 \bmod p$"

Aplicación:

Este teorema tiene importantes implicaciones en campos como la criptografía y la teoría de códigos, ya que permite generar claves seguras y verificar la integridad de los mensajes. Además, su demostración y generalizaciones han sido objeto de estudio en matemáticas avanzadas. Este teorema ha sido ampliamente utilizado en criptografía y en la teoría de números, ya que permite demostrar la primalidad de un número de manera eficiente.

Ejemplo:

Sea $p = 7$ y $a = 3$ debemos verificar si a es número primo, $a^{p-1} \equiv 1 \bmod p$

$$3^{7-1} \equiv \bmod 7$$

$$3^{6} \equiv \bmod 7$$

$$3^{3} = 27, \qquad 27 \bmod 7 = 6$$

$$(3^{3} \times 3^{3}) \bmod 7 = (6 \times 6) \bmod 7 = 36 \bmod 7 = 1$$

$$36 \equiv 1 \bmod 7$$

Se cumple con el teorema $a = 3$ es un número primo.

3. La Función Totiente de Euler $\phi(n)$:

La Función Totiente de Euler es un concepto fundamental en la teoría de números y fue introducida por el matemático suizo Leonhard Euler en el siglo XVIII. Surge como una extensión de ideas relacionadas con números primos, coprimos y propiedades modulares. Euler la definió en su trabajo sobre la teoría de congruencias y la generalización del teorema pequeño de Fermat.

Definición:

La función totiente de Euler $\phi(n)$, calcula el número de enteros positivos menores que n, que son coprimos con n.

Fórmula:

Si n es el resultado del producto de dos números primos p y q entonces:

$$\phi(n) = (p - 1)(q - 1)$$

Aplicación:

Se utiliza por ejemplo en el algoritmo RSA, para calcular la clave pública, como la clave privada, ya que está relacionada con la generación del inverso modular.

Ejemplo:

Para $p = 7$ y $q = 13$ tenemos:

$n = 7 \times 13 = 91.$

$\phi(n) = (p - 1)(q - 1)$

$\phi(n) = 72$

4. El Algoritmo Extendido de Euclides

El Algoritmo Extendido de Euclides es una mejora del Algoritmo de Euclides, utilizado para encontrar el Máximo Común Divisor (MCD) de dos números enteros. Además de calcular el

MCD, este algoritmo proporciona los coeficientes de Bézout**,** los cuales son fundamentales en muchas aplicaciones de la teoría de números y criptografía.

Este algoritmo fue desarrollado a partir del método original de Euclides, quien en su obra *Los Elementos* (300 a.C.) describió la manera de calcular el MCD de dos números mediante divisiones sucesivas. La versión extendida se formuló posteriormente y se consolidó en la matemática moderna.

Definición:

El Algoritmo Extendido de Euclides es un método eficaz para calcular:

$$\textbf{MCD}\ (a, b) = ax + by$$

Donde MCD(a, b) representa el máximo común divisor de a y b, y los coeficientes x e y son enteros que satisfacen la ecuación.

Proceso:

Dado dos números enteros a y b, el proceso sigue estos pasos:

Se debe aplicar el algoritmo de Euclides mediante divisiones sucesivas: a = bq + r donde q sería el cociente y r el residuo, este proceso se repite hasta que el residuo sea igual a cero.

Una vez encontrado el último residuo distinto de cero, este sería el MCD de a y b. Con los valores obtenidos de los residuos y cocientes anteriores, se expresa el MCD como una combinación lineal de a y b, encontrando con esto los coeficientes x e y.

Aplicaciones:

El algoritmo extendido de Euclides tiene diversas aplicaciones en la criptografía, se utiliza para encontrar el inverso de un número módulo n, lo cual es crucial en el algoritmo RSA. También Si e es una clave pública en RSA y $\phi(n)$ es la función totiente de Euler, el algoritmo extendido ayuda a calcular *d* que se utiliza en la clave privada que también es el exponente de descifrado:

Ejemplo:

Supongamos que queremos calcular el MCD(30, 18) y encontrar los coeficientes x e y.

$$30x + 18y = \text{MCD}(30, 18)$$

Lo primero a realizar es aplicar el algoritmo de Euclides:

$$30 = 18 \times 1 + 12$$
$$18 = 12 \times 1 + 6$$
$$12 = 6 \times 2 + 0$$

Como el último residuo distinto de cero es 6, tenemos que **MCD(30,18) = 6.**

Ahora debemos encontrar los coeficientes de Bézout:

Expresamos 6 en función de 30 y 18:

$$6 = 18 - 1 \times (12)$$
$$6 = 18 - 1 \times (30 - 18 \times 1)$$
$$6 = 18 - 30 + 18$$
$$6 = -30 + 2 \times 18$$
$$6 = 30\,(-1) + 18\,(2)$$

Por lo tanto, los coeficientes de x = -1 e y = 2, cumpliendo con MCD $(a, b) = ax + by$

Verificamos **MCD** $(a, b) = ax + by$

$$\mathbf{6} = 30\,(-1) + 18\,(2)$$
$$6 = -30 + 36$$
$$6 = 6$$

2.2.3 Aplicaciones de la Teoría de Números en Criptografía

La teoría de números constituye una base fundamental de la criptografía moderna, al proporcionar las herramientas matemáticas necesarias para el desarrollo de algoritmos como RSA. Sus resultados permiten trabajar con números primos, factorización y propiedades aritméticas que son esenciales en los sistemas de cifrado actuales.

Además de la aritmética modular y la teoría de números, el álgebra abstracta desempeña un papel clave en la criptografía. Sus estructuras, como grupos, anillos y cuerpos, permiten formalizar y diseñar algoritmos criptográficos eficientes, cuyos conceptos se abordarán a continuación.

2.3 Álgebra Abstracta

2.3.1 Definición

El álgebra abstracta es una rama de las matemáticas que se centra en el estudio de estructuras algebraicas como los grupos, anillos y cuerpos. A diferencia del álgebra elemental que trabaja principalmente con ecuaciones y números concretos, el álgebra abstracta se preocupa por entender las propiedades y reglas generales que subyacen en diversas operaciones matemáticas. Esta abstracción permite a los matemáticos explorar patrones comunes en una amplia gama de sistemas más allá de los números, como matrices, funciones e incluso simetrías geométricas.

2.3.2 Conceptos Claves del Álgebra Abstracta en Criptografía

El álgebra abstracta se fundamenta en la idea de estudiar estructuras matemáticas que no están limitadas a los números. Y que abarcan otros objetos como funciones, matrices y simetrías. Estas estructuras se organizan en sistemas algebraicos conocidos como grupos, anillos y cuerpos, los cuales se definen en función de ciertas reglas y propiedades comunes.

1. **Grupos**

Definición:

Un grupo es una estructura algebraica formada por un conjunto de elementos y una operación binaria (una operación que toma dos elementos y produce otro del mismo conjunto), que cumple con ciertas propiedades. Estas propiedades incluyen la asociatividad (el orden en que se agrupan los elementos no afecta el resultado), la existencia de un elemento neutro (un elemento que no cambia a otros cuando se combina con ellos) y la existencia de un inverso (un elemento que al combinarse con otro da como resultado el elemento neutro). Un ejemplo clásico de grupo es el conjunto de los números enteros con la operación de suma.

Propiedades:

1. **Clausura:** Si $a, b \in G$, entonces $a * b \in G$
2. **Asociativa:** $(a * b) * c = a * (b * c)$ para todo $a, b, c \in G$
3. **Elemento identidad:** Existe un elemento $e \in G$ tal que $a * e = e * a$ para todo $a \in G$.
4. **Inverso:** Para cada $a \in G$, existe un $b \in G$ tal que $a * b = b * a = e$.

Ejemplo:

Un ejemplo clásico de grupo es el conjunto de los números enteros Z, con la operación de suma.

1. **Clausura:** Si sumamos dos números enteros, el resultado sigue siento un número entero.
    - 5 + (-3) = 2 (como 2 ∈ Z, se cumple la Clausura).
2. **Asociativa:** La suma de enteros es asociativa, es decir:
    - (2 + 3) + 4 = 2 + (3 + 4) = 9.
3. **Elemento neutro:** El número 0 es el elemento neutro porque al sumarlo con cualquier número, el resultado sigue siendo el mismo número.
    - 7 + 0 = 7 y -4 + 0 = -4
4. **Elemento inverso:** Para cada número entero a, existe otro número $-a$ que actúa como su inverso, porque su suma da el neutro 0.
    - 6 + (-6) = 0 y -3 + 3 = 0
5. **Conmutatividad:** El orden de la suma no altera el resultado.
    - 4 + (-5) = -5 + 4 = -1

Aplicación:

En la criptografía por ejemplo se utilizan grupo cíclicos en los cifrados de Diffie-Hellman para el intercambio de claves.

2. Anillos

Definición:

Un anillo es una estructura más compleja que un grupo, ya que involucra dos operaciones: adición y multiplicación. Para que un conjunto sea considerado un anillo, la adición debe formar un grupo conmutativo y la multiplicación debe ser asociativa. Además, debe cumplirse la propiedad distributiva, que conecta ambas operaciones.

Propiedades:

1. $(\mathcal{R}, +)$ es un grupo **abeliano**, es decir, debe cumplir las cuatro propiedades de un grupo ya mencionadas y además debe cumplir con la propiedad conmutativa.
2. $(\mathcal{R}, \times)$ es cerrada, asociativa, y tiene un elemento de identidad multiplicativa.
3. La multiplicación es distributiva sobre la suma:

$$a \times (b + c) = (a \times b) + (a \times c)$$

Ejemplo:

El conjunto de los números enteros **Z** con las operaciones de suma y multiplicación cumple todas las propiedades de un anillo:

1. La adición forma un grupo conmutativo:
    - 3 + 5 = 5 + 3 = 8 (conmutatividad)
    - Existe un elemento neutro: 0, porque $a + 0 = a$.
    - Cada número tiene un inverso aditivo: $-a$, porque $a + (-a) = 0$.
2. La multiplicación es asociativa:
    - $(2 \times 3) \times 4 = 2 \times (3 \times 4) = 24$
3. La propiedad distributiva se cumple:
    - $2 \times (3 + 5) = (2 \times 3) + (2 \times 5) = 6 + 10 = 16$

Aplicaciones:

Los anillos permiten construir sistemas criptográficos seguros y eficientes. Desde RSA y curvas elípticas hasta cifrado poscuántica y computación Homomórfica, los anillos son una herramienta matemática esencial para garantizar la seguridad en la era digital. El sistema RSA se basa en anillos de enteros módulo n, donde las claves se generan a partir de la multiplicación de dos primos grandes. También lo podemos ver en la criptografía de curvas elípticas en donde los anillos permiten definir operaciones de suma y multiplicación sobre un conjunto finito de puntos en una curva.

3. Cuerpos

Un cuerpo es una estructura aún más estricta que un anillo, donde la multiplicación también forma un grupo conmutativo (exceptuando el elemento neutro de la adición, que no tiene inverso multiplicativo)

Propiedades:

Un conjunto F con las operaciones de suma + y multiplicación × es un cuerpo si cumple las siguientes propiedades:

1. **Grupo abeliano bajo la suma:** Existe un elemento neutro (0) y cada elemento tiene un inverso aditivo.
2. **Grupo abeliano bajo la multiplicación:** Existe un elemento neutro (1) y cada elemento distinto de 0 tiene un inverso multiplicativo.
3. **Distributivita:** La multiplicación es distributiva respecto a la suma.

Ejemplo:

El conjunto de los números racionales Q con las operaciones de suma y multiplicación es un cuerpo porque:

1. **Clausura:** Si sumamos o multiplicamos dos números racionales, obtenemos otro número racional.
2. **Asociatividad y conmutatividad:** Se cumple en la suma y la multiplicación.
3. **Elemento neutro:** El 0 es el neutro en la suma y el 1 en la multiplicación.
4. **Inverso aditivo:** Cada número a tiene $-a$,
5. **Inverso multiplicativo:** Todo número distinto de 0 tiene un inverso $1/a$

Aplicación:

Los cuerpos (o campos) juegan un papel fundamental en la criptografía moderna, ya que proporcionan una estructura algebraica bien definida con propiedades clave como la existencia de inversos multiplicativos, la conmutatividad y la distributividad. Estas propiedades permiten diseñar sistemas criptográficos eficientes y seguros. En las curvas elípticas se utilizan cuerpos finitos para definir operaciones seguras en un espacio matemático restringido, se realizan operaciones de suma y multiplicación sobre los puntos de la curva. En el cifrado RSA y otros algoritmos basados en números primos dependen de la aritmética en cuerpos finitos Z_p

2.3.3 Aplicaciones del álgebra abstracta en la criptografía

La criptografía moderna se apoya en la teoría de números y el álgebra abstracta para el diseño de algoritmos criptográficos seguros. En particular, las estructuras algebraicas como grupos, anillos y campos finitos permiten formalizar los procesos de cifrado, autenticación y firma digital utilizados en distintos sistemas criptográficos.

Los conceptos matemáticos desarrollados en este capítulo constituyen la base teórica necesaria para comprender los algoritmos criptográficos modernos, los cuales serán abordados con mayor detalle en el capítulo siguiente.

CAPÍTULO 3

ANÁLISIS MATEMÁTICO DEL ALGORITMO RSA

3.1 Introducción al algoritmo RSA

El algoritmo RSA es uno de los sistemas de criptografía de clave pública más utilizados en la actualidad. Su funcionamiento se basa en conceptos fundamentales de la teoría de números, como la aritmética modular y el cálculo del inverso multiplicativo, lo que permite garantizar la seguridad en los procesos de cifrado y descifrado de información.

3.2 ¿Qué es el Algoritmo RSA?

El algoritmo RSA es un sistema de cifrado asimétrico que utiliza un par de claves matemáticamente relacionadas: una clave pública para el cifrado y una clave privada para el descifrado. Cada usuario genera su propio par de claves y puede distribuir libremente la clave pública, manteniendo la confidencialidad de la información mediante la clave privada.

Elementos clave del algoritmo RSA:

El funcionamiento del algoritmo RSA se apoya en una serie de elementos fundamentales que permiten garantizar la seguridad del proceso criptográfico y comprender su análisis matemático. Entre los principales elementos se destacan los siguientes:

Claves asimétricas: RSA utiliza un par de claves matemáticamente relacionadas. La clave pública (e, n), se emplea para cifrar los mensajes, mientras que la clave privada (d, n), se utiliza para descifrarlos. La seguridad del sistema radica en que la clave privada no puede obtenerse de manera eficiente a partir de la clave pública.

Números primos: La selección de dos números primos grandes y distintos es esencial para la generación de las claves. La dificultad de factorizar el producto de estos números constituye la base de la seguridad del algoritmo.

Aritmética modular: Todas las operaciones de cifrado y descifrado se realizan mediante congruencias módulo n, lo que permite trabajar con números grandes de forma eficiente.

Exponenciación modular: El cifrado y el descifrado se realizan mediante potencias módulo n, lo que permite transformar el mensaje de forma segura sin revelar su contenido original.

Estos elementos constituyen el marco conceptual que permite comprender el desarrollo matemático del algoritmo RSA, el cual se aborda en las secciones siguientes.

3.3 Análisis del Algoritmo RSA

En esta sección se analiza un script del algoritmo RSA desarrollado con fines educativos, el cual se incluye en los anexos de esta tesis. Dicho script no representa una implementación completa de uso industrial, sino una versión simplificada diseñada para facilitar la comprensión de los conceptos matemáticos fundamentales involucrados en el algoritmo.

El análisis se centra en explicar la lógica matemática de cada etapa del proceso, desde la generación de claves hasta el cifrado y descifrado del mensaje, utilizando números pequeños que permiten visualizar claramente los cálculos realizados.

3.3.1 Generación de claves

La generación de claves es la etapa inicial y fundamental del algoritmo RSA. En este proceso intervienen conceptos matemáticos esenciales como los números primos, la función totiente de Euler y la coprimalidad.

3.3.1.1 Selección de Números Primos (*p* y *q*)

En RSA, se eligen dos números primos p y q que servirán para generar las claves. Estos números deben ser grandes y diferentes para garantizar la seguridad del sistema. Estos dos números primos se utilizan para generar la clave pública y la clave privada, y su tamaño determina el nivel de seguridad del sistema. En implementaciones reales, p y q suelen tener cientos o miles de bits, pero en este caso, trabajaremos con valores más pequeños para fines educativos.

Propiedades de *p* y *q*:

1. **Números primos:** Ambos valores deben ser primos.
2. **Diferentes:** p y q deben ser distintos para garantizar que n, el producto de p y q, sea único.
3. **Grandes:** En sistemas RSA reales, p y q son números grandes para garantizar la seguridad frente a ataques de factorización.

Verificación de Primos

El algoritmo RSA, verifica por medio de una función si los dos números elegidos son primos para mayor seguridad, en nuestro ejemplo, el script incluye la función **verifica_primo(n**), que verifica si un número dado es primo:

```
def verifica_primo(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False
    x = 3
    while x * x <= n:
        if n % x == 0:
            return False
        x += 2
    return True
```

Explicación de la función:

1. **Caso base (n < 2):**
    - Para números menores a 2, no son primos.
2. **El 2 como excepción:**
    - El 2 es el único número primo par.
3. **Números pares mayores a 2:**
    - Los números pares mayores a 2 no son primos.
4. **Verificación hasta la raíz cuadrada de n:**
    - Se evalúa si n es divisible por algún número impar desde 3 hasta $\sqrt{n}$. Si no se encuentran divisores, el número es primo.

Ejemplo:

- Entrada: n = 11
- Proceso:

- $11\%2 \neq 0$
- No hay divisores entre 3 y $\sqrt{11}$ ≈**3.32**.

- Salida: **`True`** (11 es primo).

La razón por la que se utiliza la raíz cuadrada de un número para verificar su primalidad radica en la lógica matemática de la factorización, Si un número n no es primo, significa que puede expresarse como un producto de dos factores menores que él: n = a × b.

Si a y b son divisores de n, al menos uno de ellos debe ser menor o igual que $\sqrt{n}$. De lo contrario, su producto sería mayor que n.

Por ejemplo, si tomamos 36:

$$36 = 2\times18,\ 3\times12,\ 4\times9,\ 6\times6$$

Observamos que todos los factores más pequeños están por debajo o iguales a $\sqrt{36} = 6$

Selección de *p* y *q*:

El usuario elige p y q mediante la función **pyq():**

```
def pyq():
    p = int(input("\tvalor de (p)="))
    while verifica_primo(p) == False:
        print("\t(p) tiene que ser un número primo!!!")
        p = int(input("\tvalor de (p)="))
    q = int(input("\tvalor de (q)="))
    while verifica_primo(q) == False or q == p:
        print("\t(q) tiene que ser un número primo diferente de (p)!!!")
        q = int(input("\tvalor de (q)="))
    lpq = [p, q]
    return lpq
```

Explicación del código:

1. Solicita al usuario ingresar p:
    - Si p no es primo, muestra un mensaje de error y solicita un nuevo valor.

2. Solicita q:

    - Si q no es primo o es igual a p, muestra un mensaje de error y solicita un nuevo valor.

3. Devuelve los valores de p y q.

Ejemplo Práctico: Selección de Números Primos

`Entrada del usuario:`

```
Valor de (p) = 3
Valor de (q) = 11
```

Proceso:

1. El usuario selecciona p = 3 y q = 11
2. Ambos valores son verificados como primos utilizando la función **verifica_primo**(n):

    - Para p = 3, se verifica que:

      $$\mathbf{3 \bmod 2 \neq 0}$$

      y no hay divisores hasta $\sqrt{3}$. Por lo tanto, p es primo.

    - Para q = 11, se verifica que:

      $$\mathbf{11 \bmod 2 \neq 0},\ \mathbf{11 \bmod 3 \neq 0}$$

      y no hay divisores hasta $\sqrt{11}$. Por lo tanto, q es primo.

`Salida en el Script:`

```
(p) = 3
(q) = 11
```

Con estos valores iniciales, el algoritmo continuara con el cálculo de n y $\phi(n)$.

3.3.1.2 Cálculo de n

El cálculo de n es un paso fundamental en la generación de claves del algoritmo RSA. n se utiliza como el módulo para las operaciones de cifrado y descifrado, y forma parte de las claves pública y privada.

¿Qué es n?

n es el producto de los dos números primos seleccionados (p y q). Su valor se determina:

1. **El espacio de cifrado**: n define el rango en el que pueden operar los mensajes cifrados. Todos los mensajes y resultados deben ser menores a n.
2. **La seguridad del sistema**: El nivel de seguridad depende directamente del tamaño de n. En implementaciones reales, n es el producto de dos números primos muy grandes.

En este ejemplo educativo, estamos utilizando números primos pequeños (p = 3 y q = 11) para simplificar los cálculos y la comprensión del proceso.

Fórmula para Calcular n:

$$n = p \times q$$

1. Se multiplica el número primo p por el número primo q.
2. El resultado, n, será el módulo usado para las operaciones matemáticas del algoritmo.

   Cálculo con p = 3 y q=11

$$n = 33$$

En este caso, n toma el valor de **33**.

Implementación en el Script

En el script, el cálculo de *n* se realiza de la siguiente manera:

```
n = p * q
print(f"\t(n)=(p)(q)")
print(f"\t(n)=({p})({q})")
print(f"\t(n)={n}")
```

Salida del Script:

```
(n) = (p) * (q)
(n) = (3) * (11)
(n) = 33
```

Importancia de n:

1. n asegura que las operaciones matemáticas del cifrado y descifrado se mantengan dentro de un rango finito, gracias a la **aritmética modular**.
2. En implementaciones reales, n debe ser lo suficientemente grande para prevenir ataques de factorización, donde un atacante intentaría descomponer n en sus factores *p* y *q*.

3.3.1.3 Cálculo de $\phi(n)$

El siguiente paso en la generación de claves RSA es calcular la función **totiente de Euler** $\phi(n)$\, que juega un papel crucial en la selección de la clave pública (e) y la clave privada (d).

Cálculo con $p = 3$ y $q = 11$:

1. Restamos 1 a cada número primo:

$$p - 1 = \mathbf{2}$$

$$q - 1 = \mathbf{10}$$

2. Multiplicamos los resultados:

$$\phi(n) = (p - 1) \times (q - 1)$$

$$\phi(n) = \quad 2\ 0$$

Implementación en el Script:

En el script, este cálculo se realiza de la siguiente manera:

```
φ = (p - 1) * (q - 1)
print(f"\t(φ)=(p-1)(q-1)")
print(f"\t(φ)=({p}-1)({q}-1)")
print(f"\t(φ)={φ}")
```

Salida del Script

```
(φ) = (p - 1) * (q - 1)
(φ) = (3 - 1) * (11 - 1)
(φ) = 20
```

Importancia de ϕ(n):

1. **Relación con las claves:**
    - La función $\phi(n)$\ define el rango de valores posibles para el exponente público e y garantiza la existencia del exponente privado *d*.
2. **Propiedad matemática clave:**
    - Si e es coprimo con $\phi(n)$, existe un único valor *d* que satisface $d \times e \equiv 1 \mod \phi(n)$, lo que hace posible el descifrado.

3.3.1.4 Selección del Exponente Público (e)

Una vez calculado $\phi(n)$, el siguiente paso en la generación de claves RSA es seleccionar un exponente público **(e)**. Este valor es crucial, ya que forma parte de la clave pública y se utiliza para cifrar los mensajes.

¿Qué es el Exponente Público (e)**?**

El exponente público e es un número entero que cumple dos condiciones:

1. **Debe ser coprimo con ϕ(n):**

$$\text{MCD}(e, \phi(n)) = 1$$

   Esto asegura que e y ϕ(n) no tengan factores comunes, excepto el 1.

2. **Debe estar en el rango:**

$$1 < e < \phi(n)$$

   Esto significa que e debe ser un número positivo menor que ϕ(n).

   La elección de e garantiza que existirá un valor ***d*** (exponente privado) que permita el descifrado.

Proceso de Selección de e**:**

1. Se genera una lista de posibles valores para e, que cumplan con $\text{MCD}(e, \phi(n)) = 1$.
2. El algoritmo selecciona el valor de la lista generada. Para nuestro ejemplo que estamos explicando, el usuario selecciona uno de los valores válidos.

Cálculo con $p = 3$ y $q = 11$:

- Ya calculamos que $n = 33$ y $\phi(n) = 20$
- Ahora buscamos valores para e que cumplan $1 < e < 20$ y que sean coprimos con 20.

Generación de lista de posibles valores:

- Listamos los números menores que 20.

- Eliminamos los que no son coprimos con 20

Por lo tanto, los valores posibles para e son:

$$e = \mathbf{3, 7, 9, 11, 13, 17, 19.}$$

Elección del usuario: Supongamos que el usuario elige:

$$e = \mathbf{7}$$

Implementación en el Script:

En el script, se genera automáticamente una lista de valores para e y se verifica que el valor seleccionado cumpla con MCD(e, $\phi(n)$) = 1:

En el script, este cálculo se realiza de la siguiente manera:

```
def calcula_e(phi):
    e = 2
    le = [ ]
    while e > 1 and e < phi:
        if mcd(e, phi) == 1:
            le.append(e)
        e += 1
    print("\nVALORES PARA (e)=" + str(le))
    e = int(input("\nValor de (e)="))
    while mcd(e, phi) != 1:
        print("\nElija un valor de la lista!!!")
        e = int(input("\nValor de (e)="))
    return e
```

Salida en el Script:

```
VALORES PARA (e) = [3, 7, 9, 11, 13, 17, 19]
Valor de (e) = 7
```

Importancia del Exponente Público (e):

1. **Parte de la clave pública:**
    - La clave pública se define como (n, e).
    - Esta clave se utiliza para cifrar los mensajes.
2. **Relación con el exponente privado (*d*):**
    - e debe permitir calcular *d*, el exponente privado, que es el inverso modular de e respecto a $\phi(n)$.

3.3.1.5 Cálculo del Exponente Privado (*d*)

El exponente privado (*d*) es la pieza final en la generación de claves RSA. Este valor se utiliza junto con n para descifrar mensajes cifrados con la clave pública (e, n).

¿Qué es el Exponente Privado (*d*)?

El valor *d* es el inverso modular de e respecto a $\phi(n)$. Matemáticamente, esto significa que *d* debe cumplir la siguiente ecuación:

$$d \times e \equiv 1 \bmod \phi(n)$$

Cálculo del Inverso Modular:

Para encontrar *d,* se utiliza el algoritmo extendido de Euclides, que permite resolver ecuaciones de congruencia de la forma:

$$d \times e - k \times \phi(n) = 1$$

Donde: *k* es un número entero.

Este cálculo asegura que exista una solución única para *d* cuando MCD(e, $\phi(n)$) = 1.

Cálculo con $p = 3$, $q = 11$, $e = 7$**:**

1. Ya calculamos que:

$$n = 33 \text{ y } \phi(n) = 20$$

2. El valor de e elegido es:

$$e = 7$$

3. Encontramos *d* resolviendo:

$$\boldsymbol{d \times e \equiv 1 \bmod \phi(n)}$$

$$d \cdot 7 \equiv 1 \mod 20$$

Usando el Algoritmo Extendido de Euclides:

$$\boldsymbol{d = 3}$$

Implementación en el Script:

En el script, se utiliza la función **`congruente(e, ø)`** para calcular *d*:

```
def congruente(e, phi):
    k = 1
    m = (1 + (k) * (phi)) % (e)
    while m != 0:
        k += 1
        m = (1 + (k) * (phi)) % (e)
    d = int((1 + (k) * (phi)) / (e))
    return d
```

Salida del Script:

```
(d) * (e) ≡ 1 mod (φ)
(d) * (7) ≡ 1 mod (20)
(d) = 3
```

Resultado Final:

- **Clave Pública:** $(n, e) = (33, 7)$
- **Clave Privada:** $(n, d) = (33, 3)$

Importancia del Exponente Privado (*d*):

1. Descifrado: *d* permite revertir el cifrado y recuperar el mensaje original.
2. Seguridad: La relación entre ***d*, *e* y $\phi(n)$** es la base de la seguridad de RSA, ya que calcular *d* sin conocer $\phi(n)$ requiere factorizar *n*, lo que es computacionalmente difícil para valores grandes.

3.3.2 Cifrado en el Algoritmo RSA

El cifrado es el proceso mediante el cual un mensaje original, también conocido como texto plano, se transforma en un texto cifrado, utilizando para esto la clave pública generada previamente (n, e). Con este procedimiento se asegura que solo un destinatario, aquel que posea la clave privada, pueda descifrar el mensaje y obtener el texto original.

En este punto, explicaremos como se realiza el cifrado en el algoritmo RSA, utilizando como hemos venido haciendo el Script creado para esta tesis. Es importante mencionar que, para nuestro caso explicativo, las letras del mensaje se representan con valores numéricos según la asignación A=0, B= 1, C=2, D=3 y así sucesivamente hasta Z=26, con el objetivó de simplificar los cálculos y facilitar la comprensión de los procedimientos.

El algoritmo RSA, en su implementación original, que es más robusta, utiliza los valores por ejemplo de la tabla del código ASCII para representar los caracteres del mensaje a cifrar. Cada carácter del mensaje se convierte a su valor numérico equivalente según la asignación estándar de la tabla ASCII que son valores mucho más grandes de los que vamos a utilizar en la explicación de nuestra tesis.

¿Qué es la tabla ASCII?

La Tabla ASCII (American Standard Code for Information Interchange) es un estándar de codificación de caracteres que asigna un número entero único a cada carácter, símbolo o control utilizado en sistemas computacionales. Por ejemplo:

- La letra **A** tiene el valor **65**.
- La letra **B** tiene el valor **66**.
- El carácter espacio tiene el valor **32**.

En total, la tabla ASCII define valores para 128 caracteres, que van desde el **0** al **127**.

Proceso del Cifrado RSA

El proceso del cifrado RSA es el de convertir un mensaje original "**M"**, en un mensaje cifrado "**C"**, utilizando la fórmula matemática siguiente:

$$\mathbf{C = M^{e} \mod n}$$

Donde:

- M: Es el mensaje original representado como un número entero.
- e: Es el exponente público de la clave pública (n, e).
- n: Es el módulo, calculado como $n = p \times q$, donde p y q son números primos seleccionados previamente.
- C: Es el mensaje cifrado, un número entero.

Paso a Paso del Cifrado

Paso 1: Representar el mensaje como un número entero (M)

Ejemplo: Recordar que estamos utilizando valores numéricos según la asignación A=0, B= 1, C=2, D=3 y así sucesivamente hasta Z=26 para dar valores a las letras.

Si queremos cifrar el mensaje "CASA",

- **C** tiene el valor 2,
- **A** tiene el valor 0,
- **S** tiene el valor 19,
- **A** tiene el valor 0.

Por lo tanto, el mensaje "CASA" se representa como:

M= [2, 0, 19, 0]

Paso 2: Seleccionar los números primos (*p* y *q*)

Seleccionamos:

$p = 3, q = 11$

Paso 3: Calcular n

El valor de n es el producto de *p* y *q*:

$$n = p \times q = 33$$

Paso 4: Calcular la función totiente de Euler ($\phi(n)$)

La función totiente de Euler es:

$$\phi(n) = (p-1) \times (q-1) = 20$$

Paso 5: Elegir el exponente público (e)

Elegimos e tal que $1 < e < \phi(n)$ y MCD(e, $\phi(n)$) = 1.

$$e = 3, 7, 9, 11, 13, 17, 19.$$

En este caso ejemplo elegimos:

$$e = 7$$

Paso 6: Aplicar la fórmula de cifrado

La fórmula de cifrado RSA es:

$$C = M^e \mod n$$

Donde:

- $e = 7$
- $n = 33$

El mensaje original **"CASA"** se representó como:

$$M = [2,0,19,0]$$

Procedemos a calcular el valor cifrado C para cada M

- **Cálculo para** M = 2**:**

$$C = 2^7 \bmod 33$$

Por lo tanto: **C = 29**

- **Cálculo para** M = 0

Aplicamos la fórmula:

$$C = 0^7 \bmod 33$$

$$\mathbf{C = 0}$$

- **Cálculo para** M = 19

$$C = 19^7 \bmod 33$$

Utilizamos exponenciación modular para simplificar:

1. Calculamos $19^2 \bmod 33 = 31$
2. Calculamos $19^4 \bmod 33 = 4$
3. Calculamos $19^7 \bmod 33 = 13$

Mensaje Cifrado Final

El mensaje cifrado es:

$$C = [29, 0, 13, 0]$$

Implementación del Cifrado en el Script

El cifrado en el algoritmo RSA está definido en la sección del código que maneja la función **`cifrarmensaje`** y su complemento, **`cifrarpalabra`**. Aquí es donde los valores $C = M^e \bmod n$ son calculados y almacenados.

Sección Relevante del Script

- **Función `cifrarmensaje`:** Esta función toma el mensaje completo, lo convierte a mayúsculas, y lo divide en palabras para procesarlas una por una.
- **Función `cifrarpalabra`:** Aquí es donde ocurre el cálculo clave $C = M^e \bmod n$ para cada letra del mensaje.

```
def cifrarpalabra(m, k):
    lpc = []  # Lista para almacenar los valores cifrados
    n, e = k  # Clave pública (n, e)
    cpc = ""  # Mensaje cifrado como cadena
    for i in m:
        x = buscarpos(i)  # Convertir letra a valor numérico (A=0, B=1...)
        print(f"Valor de M (letra convertida): {x}")  # Depuración
        c = pow(x, e, n)  # Exponenciación modular (C = M^e mod n)
        lpc.append(c)  # Almacenar el valor cifrado
    for k in lpc:
        cpc = cpc + str(k) + " "
    return cpc
```

- **Función `buscarpos`:** Convierte las letras en sus valores numéricos **A=0, B=1, C=2,…, Z=26.**

```
def buscarpos(x):
    alf = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    return alf.index(x)  # Devuelve la posición de la letra
```

3.3.3 Descifrado en el Algoritmo RSA

El descifrado en RSA es el proceso mediante el cual el mensaje cifrado C se convierte de nuevo en el mensaje original M, utilizando la clave privada (n, d). Este proceso asegura que únicamente el destinatario, que posee la clave privada, pueda acceder al contenido del mensaje.

Proceso del Descifrado RSA

El descifrado utiliza la fórmula matemática:

$$\mathbf{M = C^d \mod n}$$

Donde:

- M: Es el mensaje original recuperado.
- C: Es el mensaje cifrado recibido.
- d: Es el exponente privado, parte de la clave privada.
- n: Es el módulo, también compartido en la clave pública.

Paso a Paso del Descifrado

Paso 1: Identificar la clave privada

La clave privada se calcula durante el proceso de generación de claves y consiste en:

$$\text{Clave privada} = (n, d)$$

- n: Es el producto de los números primos p y q: $n = 3 \times 11 = 33$
- d: Es el inverso multiplicativo modular de $e \mod \phi(n)$. En nuestro caso, ya se calculó que $d = 3$.

Por lo tanto, la clave privada es:

$(n, d) = (33, 3)$

Paso 2: Recibir el mensaje cifrado

El mensaje cifrado C obtenido previamente fue:

$$C = [29, 0, 13, 0]$$

Este mensaje cifrado será descifrado utilizando la fórmula:

$$M = C^d \bmod n$$

Paso 3: Aplicar la fórmula de descifrado

Para cada valor en el mensaje cifrado, calculamos:

$$M = C^d \bmod n$$

- Para C = 29:

$$M = 29^3 \bmod 33$$

1. Calculamos $29^2 \bmod 33 = 16$
2. Calculamos $29^3 \bmod 33 = 2$

M = 2

- Para C = 0

$$M = 0^3 \bmod 33 = 0.$$

M = 0

- Para C = 13

$$M = 13^3 \bmod 33$$

1. Calculamos $13^2 \bmod 33 = 4$
2. Calculamos $13^3 \bmod 33 = 19$

Por lo tanto:

M = 19

- Para C = 0:

$$M = 0^3 \bmod 33 = 0.$$

$$\mathbf{M = 0}$$

Paso 4: Reconstruir el mensaje original

El mensaje descifrado es:

$$M = [2, 0, 19, 0]$$

Usamos la asignación que hemos creado para este caso A= 0, B= 1, C= 2,…, Z= 26 para reconstruir el mensaje:

- $2 \rightarrow C$,
- $0 \rightarrow A$
- $19 \rightarrow S$
- $0 \rightarrow A$

Por lo tanto, el mensaje original es:

"CASA"

Implementación en el Script

La parte del script responsable del descifrado es la función **`descifrarmensaje:`**

```
def descifrarmensaje(msj, key):
    msj = msj.upper()  # Convertir a mayúsculas
    lm = msj.split(" ")  # Dividir el mensaje cifrado por espacios
    cmc = ""  # Resultado final descifrado
    lmc = []  # Lista para almacenar palabras descifradas
    for i in lm:
        pal = descifrarnumero(i, key)  # Descifrar cada bloque usando la clave
        lmc.append(pal)
    for j in lmc:
        cmc = cmc + str(j) + " "  # Construir el mensaje descifrado
    return cmc  # Retornar el mensaje en texto claro
```

Esta función llama a **descifrarnumero**, donde se aplica la fórmula $M = C^d \mod n$:

```
def descifrarnumero(m, k):
    lmc = []  # Lista para almacenar los valores descifrados
    ln = []  # Lista para almacenar los valores numéricos del mensaje cifrado
    n, d = k  # Se asignan los valores de la clave privada (n, d)
    cnc = ""  # Variable para almacenar el mensaje descifrado final
    men = m.split(" ")  # Dividir el mensaje cifrado en partes separadas por
espacios
    for i in men:
        x = int(i)  # Convertir cada número cifrado en un entero
        ln.append(x)  # Agregarlo a la lista de valores numéricos
    for j in ln:
        m = pow(j, d, n)  # Aplicar la operación de descifrado modular
        lmc.append(m)  # Guardar el resultado descifrado en la lista
    for k in lmc:
        l = buscarlet(k)  # Convertir el número descifrado en su letra
correspondiente
        cnc = cnc + str(l)  # Construir el mensaje en texto claro
    return cnc  # Retornar el mensaje descifrado
```

El algoritmo RSA ejemplifica la aplicación directa de conceptos matemáticos en la protección de la información. A través del uso de la aritmética modular, la teoría de números y el cálculo del inverso multiplicativo, se obtiene un sistema de cifrado robusto que sustenta numerosos mecanismos de seguridad en la actualidad.

Los resultados presentados en este capítulo permiten comprender el funcionamiento del algoritmo RSA y sirven de base para las conclusiones y recomendaciones que se desarrollan a continuación.

CONCLUSIONES

Conclusiones Generales

La presente investigación permitió demostrar que la criptografía moderna no es simplemente una herramienta tecnológica, sino una construcción matemática rigurosa basada en estructuras formales bien definidas. A través del estudio histórico y teórico desarrollado en los primeros capítulos, se evidenció cómo la evolución de los sistemas de cifrado ha estado directamente ligada al progreso de la teoría de números, la aritmética modular y el álgebra abstracta.

El análisis detallado del algoritmo RSA confirmó que su seguridad no depende del secreto del procedimiento, sino de la dificultad computacional asociada al problema de factorización de números enteros grandes. Este hecho lo sitúa dentro del marco de la teoría de la complejidad computacional, donde la imposibilidad práctica de resolver ciertos problemas en tiempo polinómico garantiza la protección de la información.

Desde el punto de vista matemático, el RSA se fundamenta en:

- La estructura del grupo multiplicativo módulo n.
- La existencia del inverso modular garantizado por la condición de coprimalidad.
- La función totiente de Euler como mecanismo estructural para la construcción de claves.
- El algoritmo extendido de Euclides como herramienta esencial para calcular el exponente privado.

Asimismo, el desarrollo manual de los cálculos y su comparación con la implementación computacional permitió validar la coherencia entre la teoría matemática y su aplicación práctica, fortaleciendo el carácter formativo de esta investigación.

Se concluye que la matemática abstracta adquiere un papel central en la seguridad digital contemporánea, y que el estudio del algoritmo RSA no solo fortalece la formación teórica del estudiante de matemática, sino que también lo capacita para comprender los desafíos tecnológicos actuales, incluyendo el impacto potencial de la computación cuántica y la necesidad de desarrollar sistemas criptográficos postcuánticos.

En consecuencia, esta tesis no solo expone un algoritmo criptográfico, sino que establece un puente sólido entre la matemática pura y su aplicación en la protección de la información en el mundo digital moderno.

RECOMENDACIONES

RECOMENDACIONES

1. Se recomienda incorporar de manera formal el estudio de la criptografía moderna en asignaturas relacionadas con teoría de números, álgebra abstracta y matemática discreta dentro de la Licenciatura en Matemática.
2. Se sugiere diseñar talleres prácticos donde los estudiantes puedan implementar algoritmos como RSA mediante herramientas computacionales, fortaleciendo la conexión entre teoría y aplicación.
3. Se recomienda desarrollar investigaciones enfocadas en la seguridad de los algoritmos actuales frente a la computación cuántica, promoviendo el estudio de la criptografía postcuántica.
4. Se sugiere profundizar en el análisis de la complejidad computacional asociada a problemas como la factorización de números grandes y el logaritmo discreto.
5. Se recomienda fomentar la elaboración de artículos científicos derivados de investigaciones en criptografía matemática, fortaleciendo la producción académica institucional.
6. Se sugiere promover estudios comparativos entre diferentes algoritmos criptográficos (RSA, ECC, Diffie-Hellman), analizando sus fundamentos matemáticos y eficiencia.
7. Se recomienda incentivar el desarrollo de material didáctico adaptado para estudiantes de matemáticas, que facilite la comprensión de la criptografía desde un enfoque pedagógico.
8. Se sugiere establecer vínculos interdisciplinarios entre matemática, informática e ingeniería para ampliar el campo de aplicación de los fundamentos matemáticos estudiados.
9. Se recomienda investigar métodos eficientes para la generación segura de números primos grandes, dado su papel central en la seguridad del RSA.
10. Se sugiere promover proyectos de investigación estudiantil enfocados en la implementación y análisis de algoritmos criptográficos en entornos reales.

PERSPECTIVAS FUTURAS

Perspectivas Futuras

El desarrollo de la criptografía en las próximas décadas estará determinado por la evolución de la capacidad computacional y por los avances en tecnologías emergentes, particularmente en el ámbito de la computación cuántica. La eventual consolidación de computadoras cuánticas funcionales podría comprometer la seguridad de sistemas criptográficos tradicionales, como el algoritmo RSA, cuya robustez se fundamenta en la dificultad computacional de la factorización de números enteros grandes.

Frente a este escenario, la criptografía postcuántica se perfila como una de las áreas de investigación más relevantes dentro de la matemática aplicada y la seguridad digital. Nuevos esquemas criptográficos basados en estructuras algebraicas más complejas, retículos matemáticos, códigos correctores de errores y problemas combinatorios están siendo desarrollados con el propósito de garantizar seguridad incluso ante ataques cuánticos. Estos avances exigen un dominio más profundo de la teoría de números, el álgebra abstracta y la complejidad computacional, consolidando nuevamente el papel central de la matemática en la protección de la información.

De igual manera, se proyecta una mayor integración entre investigación matemática y desarrollo tecnológico, donde la optimización de algoritmos, la generación eficiente de números primos y el análisis formal de estructuras algebraicas seguirán siendo líneas fundamentales de estudio.

Desde la perspectiva educativa, estos cambios implican la necesidad de fortalecer la formación matemática con una visión más contextualizada y orientada a la aplicación. La enseñanza de la teoría de números, la aritmética modular y el álgebra abstracta puede enriquecerse al vincularse con problemas reales de seguridad digital, permitiendo que el estudiante comprenda no solo el rigor formal de las matemáticas, sino también su impacto directo en la sociedad contemporánea.

En este sentido, la criptografía representa un puente entre la matemática abstracta y los desafíos tecnológicos del futuro. Su estudio no solo prepara profesionales capaces de comprender sistemas de seguridad digital, sino que también impulsa el desarrollo de pensamiento lógico, análisis estructural y capacidad investigativa. Por ello, se vislumbra que la criptografía continuará consolidándose como un campo estratégico de investigación matemática, esencial para enfrentar los retos de un mundo cada vez más interconectado y digitalizado.

REFERENCIAS

REFERENCIAS

1. Calderón Sequera, R. (2023). *Fundamentos Matemáticos de la Criptografía en Clave Pública.* **https://oa.upm.es/80470/1/TFG_ROBERTO_CALDERON_SEQUERA.pdf**
2. **Castro, J., & López, J.** (2010). *Criptografía y seguridad en computación*. Alfaomega.
3. Cinvestav. (s.f.). *Análisis del cifrado Vigenère: Figura 7, tabla del alfabeto con su numeración correspondiente.* Centro de Investigación y de Estudios Avanzados del IPN. **https://delta.cs.cinvestav.mx/~francisco/cripto/VigenereAnalisis.pdf**
4. **Fernández, M., & García, L.** (2017). *Introducción a la criptografía: Fundamentos matemáticos y aplicaciones*. Ediciones Díaz de Santos.
5. **Garreta, A.** (2012). *Álgebra abstracta y sus aplicaciones en criptografía*. Ediciones UPC.
6. Gobierno de Aragón. (s. f.). *Matemáticas para la toma de decisiones.* **https://educa.aragon.es/documents/20126/2773111/%5B02.27%5D%2BMatem%C3%A1ticas%2Bpara%2Bla%2Btoma%2Bde%2Bdecisiones.pdf/c5672fa7-62c6-e9b9-722f-43a84df43f35?t=1661254615362**
7. **Gómez, J.** (2015). *Teoría de números y criptografía*. Editorial Reverté.
8. Granados Paredes, G. (2006). *Introducción a la criptografía.* Revista Digital Universitaria, 7(7). **https://www.revista.unam.mx/vol.7/num7/art55/jul_art55.pdf**
9. Khan Academy. (n.d.). *Aritmética modular.* **https://es.khanacademy.org/computing/computer-science/cryptography/modarithmetic/a/what-is-modular-arithmetic**
10. **Muñoz Muñoz, A., & Ramió Aguirre, J.** (2013). *Cifrado de las comunicaciones digitales: De la cifra clásica al algoritmo RSA* (2.ª ed.). 0xWord.
11. Ortiz Muñoz, J. J. (2009). *Criptografía y matemáticas.* Revista SUMA, (61), 17-26. **https://revistasuma.fespm.es/sites/revistasuma.fespm.es/IMG/pdf/61/017-026.pdf**
12. **Pérez, J. M.** (2019). *Fundamentos matemáticos de la criptografía para un estudiante de grado*. Ediciones Universidad de Salamanca.
13. **Rotman, J. J.** (2010). *Introducción al álgebra moderna* (3.ª ed.). Springer.
14. **Sánchez, R.** (2018). *Matemáticas para la criptografía: Aritmética modular y teoría de números*. Editorial Paraninfo.

15. Science Photo Library. (s.f.). *Alberti cipher disc* [Imagen]. Science Photo Gallery. **https://sciencephotogallery.com/featured/alberti-cipher-disc-science-photo-library.html**

16. **Singh, S.** (1999). *Los códigos secretos: La historia de la criptografía*. Ediciones Planeta.

17. **Stallings, W.** (2017). *Criptografía y seguridad de redes: Principios y práctica* (7.ª ed.). Pearson Educación.

18. StudySmarter. (s. f.). *Criptografía Algebraica: Conceptos & Ejemplos.* **https://www.studysmarter.es/resumenes/ingenieria/ingenieria-de-telecomunicaciones-ingenieria/criptografia-algebraica/**

19. Universidad Politécnica de Cartagena. (s. f.). *Nociones matemáticas para RSA.* **https://ocw.bib.upct.es/pluginfile.php/5316/mod_resource/content/1/TEMA_Matematicas_para_RSA.pdf**

ANEXO

TABLA DE VIGENÉRE

Texto Claro

Llave	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	V	w	x	y	z
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
V	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
W	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
X	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
Y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
Z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

FORMA DE USO

Ejemplo para **cifrar el mensaje** "*esto es una prueba*" con llave = "**OSCAR**", primero se toma la columna "e" del mensaje a cifrar y luego se toma la fila "o" de la llave, obteniendo la letra cifrada en la celda donde se intercepta la columna y la fila, para este caso es la letra "s"; después se continúa con la columna "s", fila "s", donde la letra que se intercepta es la "k" y así sucesivamente hasta cifrar todo el texto.

CÓDIGO FUENTE DEL ALGORITMO (SCRIPT) RSA

A continuación, se presenta el código fuente implementado en el lenguaje Python que permite realizar la generación de claves, el cifrado y el descifrado de mensajes utilizando el algoritmo RSA. Este código se ha desarrollado para fines educativos, con énfasis en la comprensión paso a paso de los cálculos matemáticos involucrados. Cabe destacar que este no representa una implementación robusta para aplicaciones industriales, pero es funcional para enseñar los conceptos fundamentales.

Estructura del Código

1. **Funciones principales:**
    - verifica_primo(n): Verifica si un número es primo.
    - genera_primos(n): Genera una lista de números primos.
    - pyq(): Permite al usuario elegir dos números primos *p* y *q*.
    - calculae(ø): Calcula el valor de e (exponente público).
    - congruente(e, ø): Calcula el valor de *d* (clave privada).
    - cifrarmensaje(msj, key): Cifra un mensaje utilizando la clave pública.
    - descifrarmensaje(msj, key): Descifra un mensaje utilizando la clave privada.
2. **Variables importantes:**
    - n: Producto de los números primos *p* y *q*, utilizado como módulo en los cálculos.
    - e: Exponente público que forma parte de la clave pública.
    - *d*: Exponente privado utilizado para descifrar mensajes.
3. **Secuencia general:**
    - Elección de números primos *p* y *q*.
    - Generación de las claves pública y privada.
    - Cifrado del mensaje original.
    - Descifrado del mensaje cifrado para recuperar el texto original.

Código Fuente

El código fuente completo es el siguiente:

```python
def verifica_primo(n):
    # Caso base: números menores a 2 no son primos
    if n < 2:
        return False
    # El 2 es el único número primo par
    if n == 2:
        return True
    # Números pares mayores que 2 no son primos
    if n % 2 == 0:
        return False
    # Verifica los divisores impares desde 3 hasta √n
    x = 3
    while x * x <= n:
        if n % x == 0:  # Si encuentra un divisor, no es primo
            return False
        x += 2  # Incrementa en 2 para omitir pares
    return True  # Si no encuentra divisores, es primo

def genera_primos(n):
    lp = []  # Lista para almacenar números primos
    x = 2    # Inicio desde el primer número primo
    while n != 0:  # Continua hasta generar "n" primos
        if verifica_primo(x):
            lp.append(x)  # Añade el primo a la lista
            n -= 1
        x += 1
    return lp


def pyq():
    p = int(input("\tValor de (p)="))  # Solicita el primer número primo
    while not verifica_primo(p):
        print("\t(p) tiene que ser un número primo válido!!!")
        p = int(input("\tValor de (p)="))

    q = int(input("\tValor de (q)="))  # Solicita el segundo número primo
    while not verifica_primo(q) or q == p:
        print("\t(q) tiene que ser un número primo diferente de (p)!!!")
        q = int(input("\tValor de (q)="))

    return [p, q]
```

```python

def calculae(ø):
    e = 2  # Inicia desde el primer valor posible para "e"
    le = []  # Lista de posibles valores de "e"
    while e > 1 and e < ø:
        if mcd(e, ø) == 1:  # Debe ser coprimo con "ø"
            le.append(e)
        e += 1
    print("\nVALORES PARA (e)=" + str(le))  # Muestra opciones válidas para "e"
    e = int(input("\n\tValor de (e)="))  # Usuario selecciona un valor
    while mcd(e, ø) != 1:
        print("\n\tElija un valor válido de la lista!!!")
        e = int(input("\n\tValor de (e)="))
    return e

def mcd(e, ø):
    while ø % e != 0:
        ø, e = e, ø % e  # Actualiza los valores de "e" y "ø"
    return e

def congruente(e, ø):
    k = 1
    while (1 + k * ø) % e != 0:  # Busca el valor de "d" tal que (d * e) ≡ 1 (mod ø)
        k += 1
    d = (1 + k * ø) // e  # Calcula "d"
    return d

def cifrarmensaje(msj, key):
    msj = msj.upper()  # Convierte el mensaje a mayúsculas
    lmc = [cifrarpalabra(char, key) for char in msj]  # Cifra cada letra del mensaje
    return " ".join(map(str, lmc))  # Une los valores cifrados como una cadena


def cifrarpalabra(m, k):
    n, e = k  # Clave pública (n, e)
    x = buscarpos(m)  # Obtiene la posición numérica de la letra
    return pow(x, e, n)  # Aplica exponenciación modular

def buscarpos(x):
    alf = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"
    return alf.index(x)

def descifrarmensaje(msj, key):
    lm = msj.split(" ")  # Divide el mensaje cifrado en números individuales
    lmc = [descifrarnumero(int(num), key) for num in lm]  # Descifra cada número
    return "".join(lmc)  # Une las letras descifradas como una cadena

def descifrarnumero(m, k):
    n, d = k  # Clave privada (n, d)
    x = pow(m, d, n)  # Aplica exponenciación modular para descifrar
    return buscarlet(x)
```

```python

def buscarlet(x):
    alf = "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"
    return alf[x]

# Flujo principal del programa
print("\t\t\t\t///////////////// METODO DE CIFRADO (RSA) ////////////////////////")

# Paso 1: Elegir números primos "p" y "q"
p, q = pyq()

# Paso 2: Calcular "n" (módulo público)
n = p * q
print("\n\t(n) = (p) * (q) = " + str(n))

# Paso 3: Calcular "ø" (totiente de Euler)
ø = (p - 1) * (q - 1)
print("\t(ø) = (p-1) * (q-1) = " + str(ø))

# Paso 4: Calcular "e" (exponente público)
e = calculae(ø)
print("\t(e) = " + str(e))

# Paso 5: Calcular "d" (clave privada)
d = congruente(e, ø)
print("\t(d) = " + str(d))


# Paso 6: Mostrar claves públicas y privadas
key_public = [n, e]
key_private = [n, d]
print("\n\tLLAVE PUBLICA=" + str(key_public) + "\n\tLLAVE PRIVADA=" + str(key_private))

# Paso 7: Cifrar un mensaje
mensaje = input("\n\tMensaje: ")
mensaje_cifrado = cifrarmensaje(mensaje, key_public)
print("\tMensaje Cifrado: " + mensaje_cifrado)

# Paso 8: Descifrar un mensaje
mensaje_descifrado = descifrarmensaje(mensaje_cifrado, key_private)
print("\tMensaje Descifrado: " + mensaje_descifrado)

```

GLOSARIO DE TÉRMINOS TÉCNICOS EN CRIPTOGRAFÍA Y RSA

A

- **Algoritmo Criptográfico:** Conjunto de reglas matemáticas utilizadas para cifrar y descifrar información.
- **Aritmética Modular:** Rama de las matemáticas que trabaja con congruencias y operaciones sobre restos en divisiones enteras.
- **Autenticación:** Proceso que verifica la identidad de un usuario o sistema.
- **Algoritmo de Shor:** Permite factorizar números enteros y calcular logaritmos discretos en tiempo polinómico, rompiendo RSA, Diffie-Hellman y ECC.
- **Algoritmo de Grover:** Acelera la búsqueda en bases de datos no estructuradas, lo que reduce la seguridad de cifrados simétricos como AES a la mitad de su fortaleza original.

B

- **Bit:** es la unidad de información más pequeña en informática y sistemas digitales. Su nombre viene de la expresión inglesa binary digit (dígito binario) y puede tener solo dos valores posibles: 0 o 1.

C

- **Cifrado:** Proceso de transformación de un mensaje original en un mensaje ilegible utilizando una clave.
- **Cifrado por Bloques:** Técnica criptográfica que divide el mensaje en bloques de tamaño fijo antes de cifrarlo.
- **Cifrado de Flujo:** Técnica de cifrado que procesa el mensaje bit a bit o carácter a carácter.
- **Cifrado Poscuántico:** es un tipo de criptografía diseñada para proteger la información contra los ataques de computadoras cuánticas, utilizando algoritmos que se consideran seguros tanto para computadoras clásicas como cuánticas

- **Clave Privada:** En criptografía asimétrica, es la clave secreta utilizada para descifrar la información.
- **Clave Pública:** En criptografía asimétrica, es la clave compartida públicamente utilizada para cifrar mensajes.
- **Código ASCII:** Sistema de codificación que asigna un valor numérico a cada carácter.
- **Colisión en Hash:** Situación en la que dos entradas diferentes generan el mismo valor hash.
- **Coprimo:** Dos números enteros a y b son coprimos (o primos entre sí) si el único divisor común que tienen es 1. Es decir, su Máximo Común Divisor (MCD) es 1.
- **Computación Homomórfica:** Es una técnica criptográfica que permite realizar operaciones matemáticas directamente sobre datos cifrados sin necesidad de descifrarlos. Esto significa que un servidor puede procesar información sin conocer su contenido
- **Criptografía:** Ciencia que estudia los métodos para proteger la información mediante técnicas de cifrado.
- **Criptografía Asimétrica:** Método criptográfico que utiliza un par de claves: una pública y una privada.
- **Criptografía Cuántica:** Rama de la criptografía que utiliza principios de la mecánica cuántica para asegurar la información.
- **Criptografía Post-Cuántica:** Es el campo de la criptografía que estudia y desarrolla algoritmos resistentes a los ataques de computadoras cuánticas. Su objetivo es diseñar sistemas criptográficos seguros incluso ante la capacidad de cómputo que podrían tener los ordenadores cuánticos en el futuro.
- **Criptografía Simétrica:** Método criptográfico en el que se usa una única clave para cifrar y descifrar.

D

- **Descifrado:** Proceso inverso al cifrado que permite recuperar el mensaje original a partir del mensaje cifrado.
- **Diffie-Hellman:** Algoritmo criptográfico para el intercambio seguro de claves en redes públicas.

E

- **Encriptación:** Sinónimo de cifrado, aunque en algunos contextos se usa de manera más informal.
- **Exponenciación Modular:** Operación matemática clave en RSA que permite calcular $C=M^e \mod n$ de manera eficiente.

F

- **Factorización de Números Primos:** Proceso de descomposición de un número en sus factores primos. Es la base de la seguridad de RSA.
- **Firma Digital:** Técnica criptográfica que permite verificar la autenticidad y la integridad de un mensaje o documento.
- **Función Hash:** Función matemática que convierte una entrada en una salida de longitud fija, utilizada en criptografía y seguridad informática.
- **Función Totiente de Euler ($\varphi(n)$)**: Función matemática que cuenta la cantidad de enteros menores que n que son coprimos con n

G

- **Generación de Claves:** Proceso en el que se seleccionan números primos y se calculan las claves pública y privada en RSA.
- **GCD (Greatest Common Divisor - Máximo Común Divisor):** Mayor número que divide exactamente a dos números.

H

- **HMAC (Hash-based Message Authentication Code):** Algoritmo que combina una función hash con una clave secreta para verificar la integridad de un mensaje.

I

- **Integridad:** Propiedad de la seguridad informática que garantiza que la información no ha sido alterada sin autorización.

K

- **Kerckhoffs' Principle:** Principio de seguridad criptográfica que establece que un sistema debe ser seguro incluso si todo acerca de él, excepto la clave, es conocido públicamente.

M

- **MAC (Message Authentication Code):** Código de autenticación de mensajes basado en una clave secreta.
- **Mensaje Cifrado:** Resultado del proceso de cifrado, un conjunto de datos que solo puede interpretarse con la clave adecuada.
- **Mensaje en Claro (Texto Plano):** Información original antes de ser cifrada.

N

- **Nonce:** Número aleatorio utilizado una sola vez en un proceso criptográfico para evitar ataques de repetición.
- **Número Primo:** Número natural mayor que 1 que solo es divisible por sí mismo y por 1.
- **Número Compuesto:** Número natural que tiene más de dos divisores.

P

- **Padding:** Técnica de seguridad utilizada para añadir datos adicionales a un mensaje antes de cifrarlo, evitando ataques de análisis de longitud.

R

- **Reducción Modular:** Proceso matemático utilizado para reducir el tamaño de un número dentro de un módulo determinado.
- **RSA (Rivest-Shamir-Adleman):** Algoritmo criptográfico asimétrico basado en la factorización de números primos.

S

- **Script:** Un conjunto de instrucciones escritas en un lenguaje de programación o secuencias automáticas que realiza una tarea específica.
- **SHA (Secure Hash Algorithm):** Familia de funciones hash seguras utilizadas en criptografía.
- **Sistema de Clave Pública:** Esquema en el que los usuarios tienen un par de claves, una pública y otra privada.

T

- **Teoría de Números:** Rama de las matemáticas que estudia propiedades de los números enteros y es utilizada en criptografía.
- **Tiempo de Computación:** Cantidad de operaciones requeridas para resolver un problema computacional en criptografía.

V

- **Vectores de Inicialización (IV - Initialization Vector):** Valores aleatorios utilizados en cifrados por bloques para aumentar la seguridad.

CRONOLOGÍA DE LA CRIPTOGRAFÍA

Antigüedad

- **Siglo V a.C.: Escítala espartana**
    - Método de cifrado por transposición utilizado por los espartanos.
    - Consistía en un bastón cilíndrico sobre el cual se enrollaba una tira de cuero con el mensaje escrito.
    - Al desenrollar la tira, el mensaje quedaba desordenado y solo podía leerse correctamente si se colocaba en un bastón del mismo diámetro.
- **Siglo II a.C.: Cifrado de Polibio**
    - Introducido por el historiador griego Polibio.
    - Utilizaba una tabla de 5x5 para convertir cada letra en una combinación de dos números.
- **Siglo I a.C.: Cifrado César**
    - Utilizado por Julio César para la comunicación segura con sus generales.
    - Consistía en un desplazamiento fijo de letras en el alfabeto (por ejemplo, con desplazamiento de 3: A → D, B → E).
    - Fue uno de los primeros métodos de cifrado por sustitución.

Edad Media y Renacimiento

- **Siglo IX: Al-Kindi y el Criptoanálisis**
    - El matemático árabe Al-Kindi desarrolló los primeros métodos para romper cifrados basados en el análisis de frecuencia.
    - Su trabajo es considerado la primera aproximación sistemática al criptoanálisis.

- **Siglo XV: Disco de Alberti**
    - Desarrollado por León Battista Alberti, considerado el "padre de la criptografía moderna".
    - Introdujo la idea de cifrados polialfabéticos con cambios de clave durante la comunicación.
    - Creó el primer cifrado que combinaba el uso de varios alfabetos de sustitución.
- **Siglo XVI: Cifrado de Vigenère**
    - Método polialfabético que utilizaba una clave para cambiar la sustitución en cada letra.
    - Fue considerado indescifrable hasta el siglo XIX cuando Charles Babbage y Friedrich Kasiski descubrieron cómo analizarlo y descifrarlo.

Siglo XIX y Principios del Siglo XX

- **Siglo XIX: Desarrollo del Criptoanálisis Moderno**
    - Charles Babbage y Friedrich Kasiski lograron descifrar el cifrado de Vigenère con el análisis de patrones en los textos cifrados.
    - Se desarrollaron los primeros estudios sistemáticos sobre la debilidad de los cifrados polialfabéticos.
- **1917: Cifrado de Vernam (One-Time Pad)**
    - Gilbert Vernam desarrolló el único sistema de cifrado teóricamente inquebrantable, el One-Time Pad.
    - La clave es aleatoria y del mismo tamaño que el mensaje, lo que impide su descifrado sin conocer la clave exacta.
- **Segunda Guerra Mundial: Máquina Enigma**
    - Utilizada por los nazis para cifrar sus comunicaciones.

- Se basaba en rotores electromecánicos que generaban una permutación compleja del alfabeto.
- Fue descifrada por Alan Turing y su equipo en Bletchley Park mediante técnicas de criptoanálisis y computación temprana, contribuyendo a la victoria aliada.

Era de la Computación y la Criptografía Moderna

- **1976: Algoritmo de Intercambio de Claves de Diffie-Hellman**
  - Whitfield Diffie y Martin Hellman introducen el primer esquema de intercambio de claves públicas.
  - Sentaron las bases de la criptografía asimétrica al permitir el intercambio seguro de claves sin necesidad de compartirlas previamente.
- **1977: Algoritmo RSA**
  - Desarrollado por Ron Rivest, Adi Shamir y Leonard Adleman.
  - Basado en la dificultad de factorizar números primos grandes.
  - Se convirtió en uno de los primeros y más utilizados métodos de cifrado asimétrico.
- **Década de 1980: Desarrollo de los Algoritmos de Hash**
  - Se popularizan las funciones hash criptográficas como MD5 y SHA-1.
  - Estas funciones permiten la verificación de integridad de datos y se utilizan en firmas digitales.
- **1990s: Estándar de Cifrado Avanzado (AES)**
  - En 1997, el NIST lanzó un concurso para reemplazar el DES debido a sus vulnerabilidades.
  - En 2001, Rijndael fue elegido como el nuevo AES, adoptado globalmente por su eficiencia y seguridad.

Criptografía en la Era Digital

- **2000s: Expansión de la Criptografía en Internet**
    - Uso de SSL/TLS para conexiones seguras en la web.
    - Aplicación de criptografía en firmas digitales, correos electrónicos y VPNs.
    - Implementación de claves públicas en transacciones bancarias y seguridad informática.
- **2010s: Blockchain y Criptomonedas**
    - Bitcoin (2009) utiliza SHA-256 y firmas digitales para su sistema descentralizado.
    - Otras criptomonedas implementan nuevos esquemas criptográficos como las curvas elípticas para mejorar la seguridad.
- **2020s: Criptografía Poscuántica**
    - Investigaciones en algoritmos resistentes a computadoras cuánticas, como lattice-based cryptography y el algoritmo de Shor.
    - Se busca reemplazar algoritmos vulnerables a la computación cuántica, como RSA y ECC.

TABLAS DE VALORES NUMÉRICOS Y CÁLCULOS PASO A PASO

Este anexo presenta las tablas con los valores numéricos utilizados en la implementación del algoritmo RSA, desde la selección de números primos hasta el cifrado y descifrado de mensajes.

1. Números Primos Utilizados

Símbolo	Descripción	Valor Utilizado
p	Primer número primo	3
q	Segundo número primo	11

2. Cálculo del Módulo *n*

Fórmula	Cálculo	Resultado
$n = p \times q$	$n = 3 \times 11$	33

3. Cálculo de la Función Totiente de Euler φ(n)

Fórmula	Cálculo	Resultado
$\varphi(n) = (p-1) \times (q-1)$	$\varphi(33) = (3-1) \times (11-1)$	20

4. Selección del Exponente Público e

Posibles valores de e	Criterio de selección	Valor escogido
3, 7, 9, 11, 13, 17, 19	Debe ser coprimo con $\varphi(n)$	3

5. Cálculo del Exponente Privado *d*

Fórmula	Cálculo	Resultado
$d \times e \equiv 1 \mod \varphi(n)$	$d \times 3 \equiv 1 \mod 20$	7

6. Claves Generadas

Clave	Valores
Clave Pública	$(n, e) = (33, 3)$
Clave Privada	$(n, d) = (33, 7)$

7. Cifrado del Mensaje

Ejemplo: Cifrar la letra **"S"** (Valor 19)

Fórmula	Cálculo	Resultado
$C = M^e \mod n$	$C = 19^3 \mod 33$	28

Mensaje cifrado: **28**

8. Descifrado del Mensaje

Ejemplo: Descifrar C = 28

Fórmula	Cálculo	Resultado
$M = C^d \mod n$	$M = 28^7 \mod 33$	19

Mensaje descifrado: **S** (se recupera correctamente).

TABLA DE CONVERSIÓN DEL MENSAJE A NÚMEROS

En este anexo se presenta una tabla que muestra la conversión de cada letra del alfabeto en su respectivo valor numérico. Este método se ha utilizado en la implementación del algoritmo RSA dentro de la presente tesis, facilitando la comprensión y aplicación del cifrado y descifrado de mensajes.

Tabla de Conversión

Cada letra del alfabeto se asigna a un valor numérico comenzando desde **A = 0** hasta **Z = 26**.

Letra	Valor Numérico	Letra	Valor Numérico	Letra	Valor Numérico
A	0	J	9	S	19
B	1	K	10	T	20
C	2	L	11	U	21
D	3	M	12	V	22
E	4	N	13	W	23
F	5	Ñ	14	X	24
G	6	O	15	Y	25
H	7	P	16	Z	26
I	8	Q	17		

TABLA DE CONVERSIÓN ASCII

Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII	Carácter	Valor ASCII
A	65	B	66	C	67	D	68
E	69	F	70	G	71	H	72
I	73	J	74	K	75	L	76
M	77	N	78	O	79	P	80
Q	81	R	82	S	83	T	84
U	85	V	86	W	87	X	88
Y	89	Z	90	a	97	b	98
c	99	d	100	e	101	f	102
g	103	h	104	i	105	j	106
k	107	l	108	m	109	n	110
o	111	p	112	q	113	r	114
s	115	t	116	u	117	v	118
w	119	x	120	y	121	z	122
0	48	1	49	2	50	3	51
4	52	5	53	6	54	7	55
8	56	9	57	.	46	,	44
:	58	;	59	!	33	?	63
@	64	#	35	$	36	%	37
&	38	(	40	)	41	_	95
+	43	-	45	=	61	/	47

Explicación

- En el algoritmo RSA estándar, los caracteres del mensaje se convierten a sus valores ASCII antes de realizar el cifrado.
- Al descifrar, se obtiene el número correspondiente y se usa esta tabla para convertirlo nuevamente en el carácter original.

DIAGRAMA DE FLUJO DEL ALGORITMO RSA

1. **Inicio**
    - Iniciar el proceso de generación de claves.
2. **Generación de Claves**
    - **Paso 1:** Seleccionar dos números primos grandes p y q.
    - **Paso 2:** Calcular $n = p \times q$.
    - **Paso 3:** Calcular $\phi(n) = (p-1) \times (q-1)$.
    - **Paso 4:** Seleccionar un número e tal que $1 < e < \phi(n)$ y mcd $(e, \phi(n)) = 1$.
    - **Paso 5:** Calcular d como el inverso modular de e módulo $\phi(n)$, es decir, $d \times e \equiv 1 \mod \phi(n)$.
    - **Paso 6:** La clave pública es (e, n).
    - **Paso 7:** La clave privada es (d, n).
3. **Cifrado**
    - **Paso 1:** Obtener la clave pública (e, n).
    - **Paso 2:** Convertir el mensaje M en un número m tal que $0 \leq m < n$.
    - **Paso 3:** Calcular el texto cifrado $c = m^e \mod n$.
4. **Descifrado**
    - **Paso 1:** Obtener la clave privada (d, n).
    - **Paso 2:** Calcular el mensaje original $m = c^d \mod n$.
    - **Paso 3:** Convertir mm de nuevo a texto plano M.
5. **Fin**
    - Finalizar el proceso.

Explicación de los Pasos

- **Generación de Claves:**
    - Se eligen dos números primos grandes p y q.
    - Se calcula n como su producto, que será parte de ambas claves.
    - $\phi(n)$ es la función de Euler, que cuenta los números coprimos con n.
    - e es un número coprimo con $\phi(n)$, y d es su inverso modular.

- **Cifrado:**
    - El mensaje M se convierte en un número m y se cifra usando la clave pública (e, n).
- **Descifrado:**
    - El texto cifrado c se descifra usando la clave privada (d, n) para recuperar el mensaje original m.