COLOMBIA POTENCIA DE LA VIDA

TIC

Sistemas RSA de claves **privadas y públicas**

Lección 1- Módulo 2 - Unidad 2

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Desarrollo de la sesión:

En un mundo cada vez más digitalizado y conectado, la seguridad de la información se vuelve un tema crucial. Los datos sensibles circulan por redes y dispositivos, y su protección se vuelve indispensable para evitar robos, fraudes y otros delitos informáticos. En este contexto, los sistemas RSA de claves privadas y públicas emergen como pilares fundamentales en la protección de la información.

Su objetivo principal es garantizar la seguridad de las comunicaciones digitales y la integridad de los datos. Los sistemas RSA se basan en un par de claves, una pública y otra privada, matemáticamente relacionadas. La clave pública se utiliza para encriptar información, mientras que la clave privada se utiliza para desencriptarla. Esta asimetría es lo que hace que los sistemas RSA sean tan seguros.

Para comprender cómo funcionan los sistemas RSA, es necesario conocer los principios básicos de la criptografía. La encriptación es el proceso de convertir información legible en un código ininteligible para que solo las personas con la clave adecuada puedan leerla. La firma digital es un método para verificar la autenticidad de un mensaje o documento electrónico y garantizar que no haya sido modificado desde que se creó.

+ info

Los **sistemas RSA** se utilizan en una amplia variedad de aplicaciones de seguridad informática. Algunos ejemplos incluyen:

- **Correo electrónico seguro:** Los **sistemas RSA** se pueden utilizar para encriptar correos electrónicos y protegerlos de miradas indiscretas.
- **Transacciones en línea:** Los **sistemas RSA** se utilizan para proteger las transacciones en línea, como las compras con tarjeta de crédito.
- **Firmas digitales:** Los **sistemas RSA** se pueden utilizar para crear firmas digitales que garanticen la autenticidad de documentos electrónicos.

La seguridad de los sistemas RSA depende de la gestión adecuada de las claves. Las claves privadas deben mantenerse en secreto y seguras, mientras que las claves públicas pueden ser compartidas públicamente. Es importante implementar medidas de seguridad para proteger las claves y evitar que sean robadas o interceptadas.

La evaluación y la mejora continua son fundamentales para mantener la seguridad de los sistemas RSA. Es importante realizar auditorías de seguridad regulares para identificar y mitigar las vulnerabilidades. También es importante mantenerse al día con las últimas amenazas y tendencias en seguridad informática.

Los sistemas RSA, considerados un pilar de la criptografía moderna, se basan en un ingenioso esquema de criptografía asimétrica que utiliza dos claves matemáticamente relacionadas: una clave pública y una clave privada.

1. Generación de Claves:

Números primos: La base del sistema RSA reside en la selección de dos números primos grandes, p y q. Estos números deben ser seleccionados cuidadosamente, ya que su tamaño y aleatoriedad determinan la seguridad del sistema.

Cálculo del módulo: A partir de p y q, se calcula el módulo n, que es el producto de ambos números primos. Este módulo juega un papel fundamental en las operaciones de cifrado y descifrado.

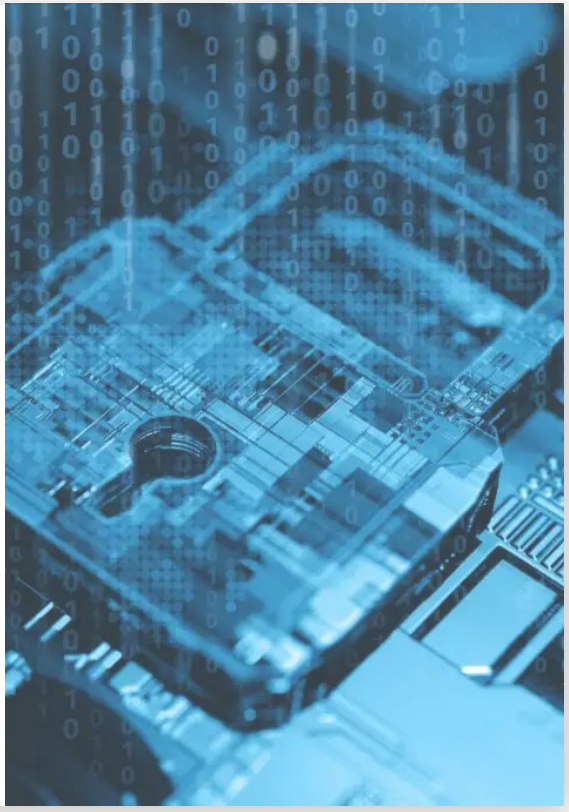
Exponente público: Se elige un número entero e que sea coprimo con (p-1)(q-1). Este valor se conoce como el exponente público y se hace público para que cualquier persona pueda utilizarlo para cifrar mensajes.

Exponente privado: Se calcula un segundo exponente, d, que es el inverso modular de e módulo (p-1)(q-1). Este valor, conocido como el exponente privado, se mantiene en secreto y es fundamental para descifrar mensajes.

2. Cifrado y Descifrado:

Cifrado: Para enviar un mensaje confidencial, el emisor utiliza la clave pública del receptor (e, n) para cifrar el mensaje. El proceso de cifrado implica elevar cada letra del mensaje a la potencia e y módulo n. El resultado es un conjunto de números que forman el criptograma.

Descifrado: El receptor, utilizando su clave privada (d, n), puede descifrar el criptograma. El proceso de descifrado implica elevar cada número del criptograma a la potencia d y módulo n. El resultado es el mensaje original en texto plano.



Seguridad:
La seguridad de los sistemas RSA depende de la dificultad de factorizar el módulo n. Si un atacante logra factorizar n, podría obtener la clave privada a partir de la clave pública. Por lo tanto, la elección de números primos grandes y la protección de la clave privada son esenciales para mantener la seguridad del sistema.

Aplicaciones:
Los sistemas RSA se utilizan en una amplia gama de aplicaciones, incluyendo:

Correo electrónico seguro:
Permite enviar y recibir mensajes confidenciales por correo electrónico.

Transacciones en línea:
Protege las transacciones financieras en línea, como las compras con tarjeta de crédito.

Firmas digitales:
Permite firmar documentos electrónicos de forma segura y verificar su autenticidad.

COLOMBIA POTENCIA DE LA VIDA

TIC

Conceptos Clave:

- Criptografía asimétrica
- Clave pública
- Clave privada
- Números primos
- Factorización
- Módulo
- Exponente público
- Exponente privado
- Cifrado
- Descifrado

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

Universidad Tecnológica de Pereira

Ejemplo 1: Generación de Claves RSA

Paso 1: Generación de Claves

Supongamos que elegimos dos números primos para nuestro ejemplo: **p = 17 y q = 11.**

1. Calculamos **N = p × q = 17 × 11 = 187.**
2. Calculamos **ϕ(N) = (p-1) × (q-1) = 16 × 10 = 160.**

Elección de la Clave Pública:

Tomemos **e = 7**. Como e debe ser relativamente primo **a ϕ(N)**, y **7** y **160** son coprimos, podemos usar **e = 7** como nuestra clave pública.

Cálculo de la Clave Privada:

Para calcular **d**, el inverso multiplicativo de e módulo **ϕ(N)**, usamos el algoritmo de Euclides extendido:

$$7d \equiv 1 \pmod{160}$$

Resolviendo esta congruencia, encontramos que **d = 23.**

Elección de la Clave Pública:

Para verificar que nuestras claves son válidas, comprobamos que $e \times d \equiv 1 \pmod{\phi(N)}$:

$$7 \times 23 \equiv 1 \pmod{160}$$
$$161 \equiv 1 \pmod{160}$$

La congruencia se cumple, por lo que nuestras claves son válidas.

Ejemplo 1: Generación de Claves RSA

Cifrado del Mensaje:

Primero, convertimos el mensaje **"HELLO"** en valores numéricos utilizando la codificación ASCII:

- **H: 72**
- **E: 69**
- **L: 76**
- **L: 76**
- **O: 79**

Elección de la Clave Pública:

Luego, ciframos cada valor numérico utilizando la clave pública **e = 7 y N = 187**:

+ info

C = M^e mod N

Para "H":

C = 72^7 mod 187 = 11

Para "E":

C = 69^7 mod 187 = 101

Para "L":

C = 76^7 mod 187 = 59

Para "L":

C = 76^7 mod 187 = 59

Para "O":

C = 79^7 mod 187 = 103

Desciframos cada valor numérico:

Para "11":
M = 11^23 mod 187 = 72

Para "101":
M = 101^23 mod 187 = 69

Para "59":
M = 59^23 mod 187 = 76

Para "59":
M = 59^23 mod 187 = 76

Para "103":
M = 103^23 mod 187 = 79

Por lo tanto, el mensaje descifrado es "HELLO".

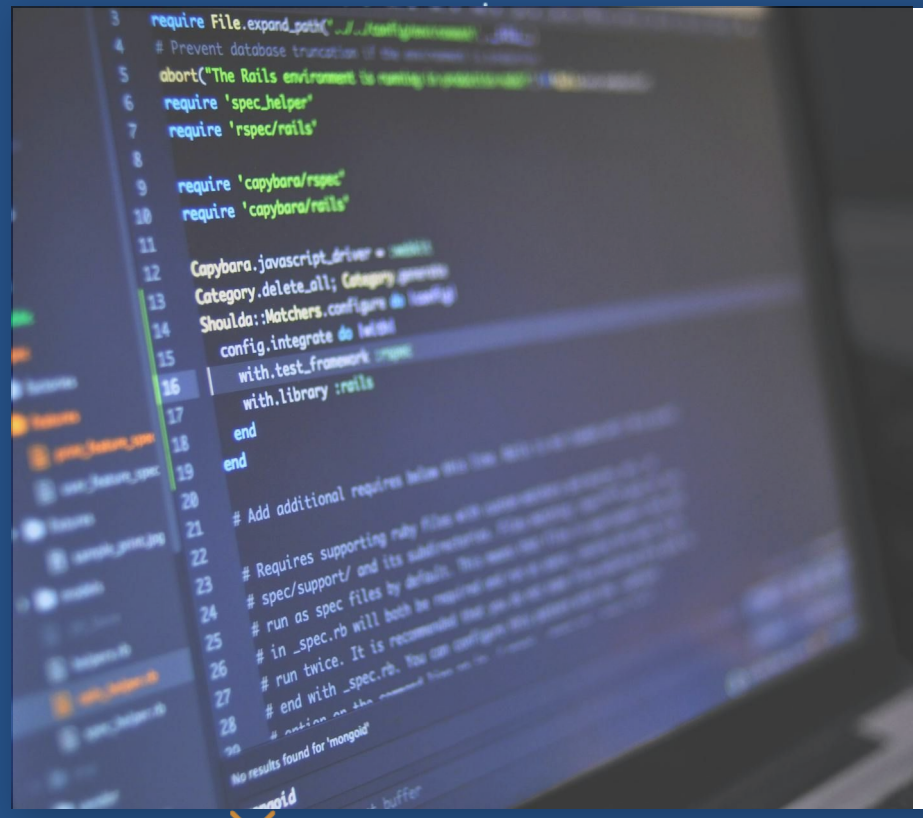
Ahora, usamos python para la operación de cifrar un mensaje utilizando RSA

```
# Función para cifrar un mensaje utilizando RSA
def cifrar_rsa(mensaje, e, N):
    mensaje_cifrado = [pow(ord(caracter), e, N) for caracter in mensaje]
    return mensaje_cifrado

# Función para descifrar un mensaje cifrado utilizando RSA
def descifrar_rsa(mensaje_cifrado, d, N):
    mensaje_descifrado = [pow(valor, d, N) for valor in mensaje_cifrado]
    mensaje_descifrado_caracteres = ''.join([chr(valor) for valor in mensaje_descifrado])
    return mensaje_descifrado_caracteres
```


```
# Función para cifrar un mensaje utilizando RSA
def cifrar_rsa(mensaje, e, N):
    mensaje_cifrado = [pow(ord(caracter), e, N) for caracter in mensaje]
    return mensaje_cifrado

# Función para descifrar un mensaje cifrado utilizando RSA
def descifrar_rsa(mensaje_cifrado, d, N):
    mensaje_descifrado = [pow(valor, d, N) for valor in mensaje_cifrado]
    mensaje_descifrado_caracteres = ''.join([chr(valor) for valor in mensaje_descifrado])
    return mensaje_descifrado_caracteres
```

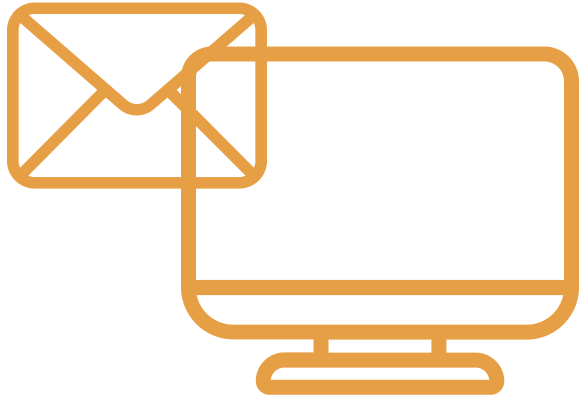


Aplicaciones Prácticas de los Sistemas RSA:

Los sistemas RSA, conocidos por su robustez y seguridad, encuentran numerosas aplicaciones en el mundo digital moderno. A continuación, se detallan algunas de las aplicaciones más destacadas de los sistemas RSA y cómo contribuyen a garantizar la seguridad y privacidad de la información:

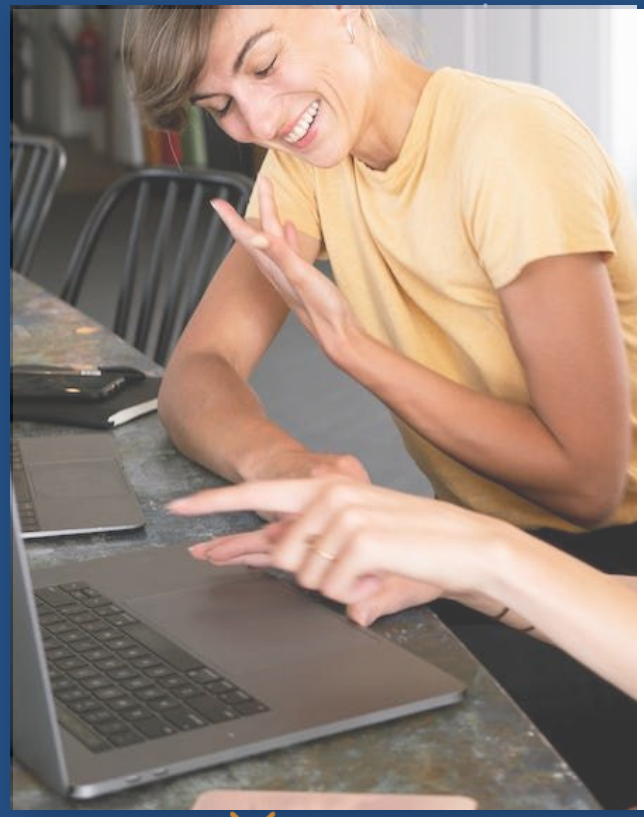
1.Cifrado de Correos Electrónicos:

El correo electrónico es una forma común de comunicación en la era digital, pero también puede ser vulnerable a la interceptación por parte de terceros malintencionados. Los sistemas RSA se utilizan para cifrar correos electrónicos, protegiendo así la confidencialidad de los mensajes durante la transmisión.



Explicación:

Cuando un usuario envía un correo electrónico cifrado utilizando RSA, el mensaje se codifica con la clave pública del destinatario. Solo el destinatario, que posee la clave privada correspondiente, puede descifrar el mensaje. Esto garantiza que solo el destinatario previsto pueda leer el contenido del correo electrónico, incluso si es interceptado durante la transmisión.



Desarrollo y explicación:

RSA es un algoritmo de cifrado asimétrico ampliamente utilizado para proteger la confidencialidad y la integridad de los datos en la comunicación digital. En RSA, cada entidad tiene un par de claves: una clave pública que se comparte con otros para cifrar mensajes, y una clave privada que se mantiene en secreto y se utiliza para descifrar los mensajes cifrados.

En este ejemplo, utilizaremos Python para generar un par de claves RSA, cifrar un mensaje utilizando la clave pública y luego descifrar el mensaje cifrado utilizando la clave privada.

+ info

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
```

```python
# Función para generar un par de claves RSA
def generar_claves():
    private_key = rsa.generate_private_key(
        public_exponent=65537, # Exponente público recomendado
        key_size=2048 # Longitud de la clave
    )
    public_key = private_key.public_key()
    return private_key, public_key
```

```python
# Función para cifrar un mensaje utilizando RSA
def cifrar_mensaje(mensaje, clave_publica):
    cifrado = clave_publica.encrypt(
        mensaje.encode(), # Exponente público recomendado
        padding.OAEP( # Relleno de texto plano óptimo (OAEP)
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return cifrado
```

```python
# Función para descifrar un mensaje cifrado utilizando RSA
def descifrar_mensaje(mensaje_cifrado, clave_privada):
    mensaje_descifrado = clave_privada.decrypt(
        mensaje_cifrado,
        padding.OAEP(  # Utilizar el mismo relleno que en el cifrado
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return mensaje_descifrado.decode()  #Convertir los bytes descifrados a cadena
```

```
# Generar un par de claves RSA
clave_privada, clave_publica = generar_claves()

# Mensaje a cifrar
mensaje_original = "Este es un mensaje confidencial"

# Cifrar el mensaje utilizando la clave pública del destinatario
mensaje_cifrado = cifrar_mensaje(mensaje_original, clave_publica)

# Descifrar el mensaje cifrado utilizando la clave privada del destinatario
mensaje_descifrado = descifrar_mensaje(mensaje_cifrado, clave_privada)

# Imprimir los resultados
print("Mensaje original:", mensaje_original)
print("Mensaje cifrado:", mensaje_cifrado)
print("Mensaje descifrado:", mensaje_descifrado)
)
```

2. Firma Digital:

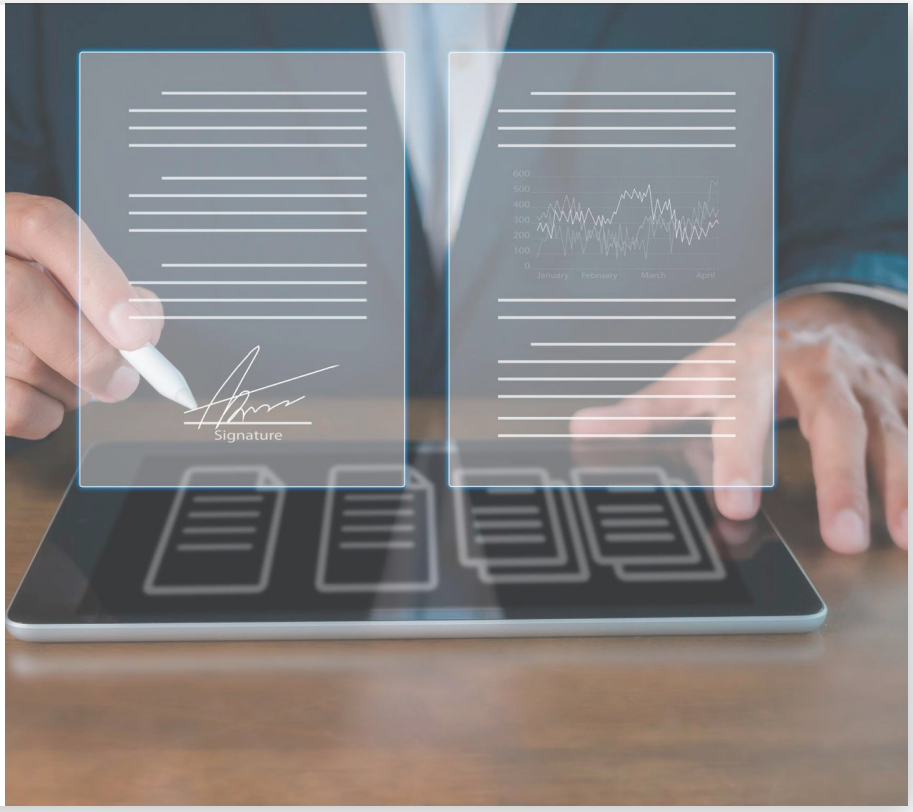
La firma digital es una técnica utilizada para autenticar la identidad del remitente y garantizar la integridad del contenido de un documento digital o una transacción en línea. Los sistemas RSA son fundamentales para generar firmas digitales seguras.

Explicación:

Para crear una firma digital, el remitente utiliza su clave privada para cifrar un resumen (hash) del documento o mensaje. El destinatario puede verificar la firma utilizando la clave pública del remitente y comparando el hash descifrado con un nuevo cálculo del hash del documento. Si los valores coinciden, la firma se considera válida y se puede verificar la autenticidad del remitente y la integridad del contenido.

Actividad:

Los estudiantes pueden participar en una actividad de simulación de firma digital. Se les puede proporcionar un documento digital y se les pedirá que generen una firma digital utilizando sus claves privadas. Luego, pueden intercambiar documentos y verificar las firmas digitales de sus compañeros utilizando las claves públicas correspondientes.



Desarrollo y explicación:

En esta actividad de simulación de firma digital, los estudiantes aprenderán cómo utilizar el algoritmo RSA para firmar digitalmente documentos y verificar las firmas digitales de sus compañeros. Utilizaremos Python y la biblioteca cryptography para llevar a cabo esta actividad.

+ info

```
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
```

```python
# Función para generar un par de claves RSA
def generar_claves():
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048
    )
    public_key = private_key.public_key()
    return private_key, public_key
```

```python
# Función para firmar un documento digital utilizando RSA
def firmar_documento(documento, clave_privada):
    firma = clave_privada.sign(
        documento,
        padding.PSS(
            mgf=padding.MGF1(hashes.SHA256()),
            salt_length=padding.PSS.MAX_LENGTH
        ),
        hashes.SHA256()
    )
    return firma
```

```python
# Función para verificar una firma digital utilizando RSA
def verificar_firma(documento, firma, clave_publica):
    try:
        clave_publica.verify(
            firma,
            documento,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        return True
    except:
        return False
```

```
# Generar un par de claves RSA para cada estudiante
clave_privada_alice, clave_publica_alice = generar_claves()
clave_privada_bob, clave_publica_bob = generar_claves()

# Documento a firmar
documento = b"Este es un documento confidencial."

# Alice firma el documento
firma_alice = firmar_documento(documento,
clave_privada_alice)

# Bob verifica la firma de Alice
verificacion_bob =
verificar_firma(documento, firma_alice,
clave_publica_alice)

# Imprimir resultados
print("Documento:", documento)
print("Firma de Alice:", firma_alice)
print("Verificación de Bob:", verificacion_bob)
```

+ info

Primero, se genera un par de claves RSA para cada estudiante utilizando la función generar_claves(). Luego, se crea una función firmar_documento() para que cada estudiante firme digitalmente el documento proporcionado utilizando su clave privada. Posteriormente, se implementa una función verificar_firma() para que los compañeros puedan verificar la firma digital utilizando la clave pública del firmante. Finalmente, se prueba el proceso firmando un documento como Alice y verificando la firma como Bob.

Identificación y Mitigación de Vulnerabilidades en Sistemas RSA

En el contexto de la seguridad de la información, los sistemas RSA representan un pilar fundamental en la protección de datos sensibles mediante la criptografía asimétrica. Sin embargo, es crucial reconocer que, a pesar de su reputación de robustez, los sistemas RSA no están exentos de vulnerabilidades potenciales. En este sentido, es fundamental que los estudiantes adquieran un entendimiento profundo de estos riesgos y aprendan estrategias efectivas para mitigarlos.

Factorización de Claves:
La seguridad de RSA se basa en la premisa de que la factorización de claves es un problema computacionalmente difícil. Los estudiantes explorarán este concepto, comprendiendo cómo la seguridad del sistema depende de la dificultad de descomponer un número compuesto en sus factores primos. Esta discusión resalta la importancia de generar claves con números primos extremadamente grandes para garantizar la resistencia a los ataques de factorización.

Ataques de Fuerza Bruta:
Otro riesgo significativo que enfrentan los sistemas RSA son los ataques de fuerza bruta, donde los adversarios intentan descifrar mensajes probando todas las combinaciones posibles de claves. Se explicará cómo el aumento en la longitud y complejidad de las claves puede dificultar significativamente estos ataques, proporcionando una capa adicional de seguridad para los sistemas RSA.