



# Guía para la Realización del Ejercicio sobre RSA y OpenSSL

## Introducción al Algoritmo RSA

RSA es un algoritmo de cifrado asimétrico ampliamente usado en seguridad digital. Este algoritmo, desarrollado en 1977 por Rivest, Shamir y Adleman, permite cifrar y firmar mensajes mediante dos claves: una clave pública y una clave privada. El procedimiento RSA se basa en la dificultad de factorizar grandes números primos, lo que proporciona seguridad en las comunicaciones cifradas y en la autenticidad de los mensajes. OpenSSL implementa RSA mediante herramientas como `pkeyutl` y `rsautl`, permitiendo ejecutar operaciones de cifrado, descifrado, firmado y verificación de mensajes con RSA.

## Ejercicio: Realización del Ejemplo con OpenSSL

Para realizar el ejercicio, siga los pasos a continuación junto a un compañero. Usarán OpenSSL para simular una transmisión segura de un mensaje firmado y cifrado.

### Requisitos previos

1. **Instalar OpenSSL** en ambos equipos.
2. **Configurar Telegram o un medio para compartir archivos** de manera segura.

### Pasos a seguir

#### 1. Generación de Claves

Cada usuario generará su par de claves RSA (pública y privada).

- Usuario Emisor:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out  
privkey-userS.pem  
openssl pkey -in privkey-userS.pem -out pubkey-userS.pem -pubout
```

- Usuario Receptor:

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out  
privkey-userR.pem  
openssl pkey -in privkey-userR.pem -out pubkey-userR.pem -pubout
```

**Nota:** Asegurarse de intercambiar las claves públicas generadas ( `pubkey-userS.pem` y `pubkey-userR.pem` ).

## 2. Emisión del Mensaje

1. Crear un archivo con el mensaje a enviar:

```
echo "Contenido del mensaje" > message-userS.txt
```

2. **Firmar el mensaje** utilizando la clave privada del emisor:

```
openssl dgst -sha256 -sign privkey-userS.pem -out  
message-userS.txt.sgn message-userS.txt
```

3. **Cifrar el mensaje** con la clave pública del receptor:

```
openssl pkeyutl -encrypt -in message-userS.txt -pubin -inkey  
pubkey-userR.pem -out message-userS.txt.enc
```

4. Enviar al receptor los archivos `message-userS.txt.sgn` (firma) y `message-userS.txt.enc` (mensaje cifrado).

## 3. Recepción del Mensaje

1. **Descifrar el mensaje** usando la clave privada del receptor:

```
openssl pkeyutl -decrypt -in message-userS.txt.enc -inkey  
privkey-userR.pem -out rec-message-userS.txt
```

2. **Verificar la firma** para asegurar que el mensaje fue enviado por el emisor legítimo:

```
openssl dgst -sha256 -verify pubkey-userS.pem -signature  
message-userS.txt.sgn rec-message-userS.txt
```

Si el mensaje de verificación es `Verified OK`, el receptor puede estar seguro de la autenticidad del mensaje y de su integridad.

## Notas Importantes

- **Clave privada y pública:** La seguridad del ejercicio depende de mantener las claves privadas seguras y confidenciales.
- **Formato de cifrado:** RSA es óptimo para mensajes cortos. Para mensajes más largos, considera usar cifrado simétrico en combinación con RSA para compartir la clave simétrica.
- **Uso de hash:** El algoritmo SHA-256 se usa para crear una huella única del mensaje.