

# LABORATÓRIO – CRIPTOGRAFIA DE DADOS (SIMÉTRICA, ASSIMÉTRICA), ASSINATURA DIGITAL:

Nome: Enzo Arrue Juan Fuso

## Objetivos

Parte 1: Criptografia Simétrica usando o OpenSSL (Algoritmos DES e AES)

Parte 2: Criptografia Assimétrica usando o OpenSSL (Algoritmos RSA)

Parte 3: Gerando o Hash do arquivo e Assinando um arquivo (Algoritmo RSA e Hash)

**OpenSSL (Linux):** é uma implementação de código aberto dos protocolos SSL e TLS. A biblioteca (escrita na linguagem C) implementa as funções básicas de criptografia e disponibiliza várias funções utilitárias. Pode ser também utilizada para assinaturas digitais e criação de certificados digitais. O OpenSSL está disponível para a maioria dos sistemas do tipo Unix, incluindo Linux, MacOS, BSD e Windows.

## Formulário:

### Parte 1: Criptografia Simétrica usando o OpenSSL (Algoritmos DES e AES)

```
(root@kali) ~ [~/home/kali]
# openssl ciphers -v
TLS_AES_256_GCM_SHA384 TLSv1.3 Kx=any Au=any Enc=AESGCM(256) Mac=AEAD
TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any Au=any Enc=CHACHA20/POLY1305(256) Mac=AEAD
TLS_AES_128_GCM_SHA256 TLSv1.3 Kx=any Au=any Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD
DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=DSS Enc=AESGCM(256) Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=ECDSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=ECDH Au=RSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
DHE-RSA-CHACHA20-POLY1305 TLSv1.2 Kx=DH Au=RSA Enc=CHACHA20/POLY1305(256) Mac=AEAD
ECDHE-ECDSA-AES256-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(256) Mac=AEAD
DHE-RSA-AES256-CCM TLSv1.2 Kx=DH Au=RSA Enc=AESCCM(256) Mac=AEAD
ECDHE-ECDSA-ARIA256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=ARIAGCM(256) Mac=AEAD
ECDHE-ARIA256-GCM-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=ARIAGCM(256) Mac=AEAD
DHE-DSS-ARIA256-GCM-SHA384 TLSv1.2 Kx=DH Au=DSS Enc=ARIAGCM(256) Mac=AEAD
DHE-RSA-ARIA256-GCM-SHA384 TLSv1.2 Kx=DH Au=RSA Enc=ARIAGCM(256) Mac=AEAD
ADH-AES256-GCM-SHA384 TLSv1.2 Kx=DH Au=None Enc=AESGCM(256) Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(128) Mac=AEAD
DHE-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AESGCM(128) Mac=AEAD
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES128-CCM TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM(128) Mac=AEAD
DHE-RSA-AES128-CCM TLSv1.2 Kx=DH Au=RSA Enc=AESCCM(128) Mac=AEAD
ECDHE-ECDSA-ARIA128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=ARIAGCM(128) Mac=AEAD
ECDHE-ARIA128-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=ARIAGCM(128) Mac=AEAD
DHE-DSS-ARIA128-GCM-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=ARIAGCM(128) Mac=AEAD
DHE-RSA-ARIA128-GCM-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=ARIAGCM(128) Mac=AEAD
ADH-AES128-GCM-SHA256 TLSv1.2 Kx=DH Au=None Enc=AESGCM(128) Mac=AEAD
ECDHE-ECDSA-AES256-CCM8 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM8(256) Mac=AEAD
ECDHE-ECDSA-AES128-CCM8 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AESCCM8(128) Mac=AEAD
DHE-RSA-AES256-CCM8 TLSv1.2 Kx=DH Au=RSA Enc=AESCCM8(256) Mac=AEAD
DHE-RSA-AES128-CCM8 TLSv1.2 Kx=DH Au=RSA Enc=AESCCM8(128) Mac=AEAD
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(256) Mac=SHA384
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(256) Mac=SHA384
DHE-RSA-AES256-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(256) Mac=SHA256
DHE-DSS-AES256-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AES(256) Mac=SHA256
ECDHE-ECDSA-CAMELLIA256-SHA384 TLSv1.2 Kx=ECDH Au=ECDSA Enc=Camellia(256) Mac=SHA384
ECDHE-RSA-CAMELLIA256-SHA384 TLSv1.2 Kx=ECDH Au=RSA Enc=Camellia(256) Mac=SHA384
DHE-RSA-CAMELLIA256-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=Camellia(256) Mac=SHA256
DHE-DSS-CAMELLIA256-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=Camellia(256) Mac=SHA256
ADH-AES256-SHA256 TLSv1.2 Kx=DH Au=None Enc=AES(256) Mac=SHA256
ADH-CAMELLIA256-SHA256 TLSv1.2 Kx=DH Au=None Enc=Camellia(256) Mac=SHA256
ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=AES(128) Mac=SHA256
ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AES(128) Mac=SHA256
DHE-RSA-AES128-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=AES(128) Mac=SHA256
DHE-DSS-AES128-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=AES(128) Mac=SHA256
ECDHE-ECDSA-CAMELLIA128-SHA256 TLSv1.2 Kx=ECDH Au=ECDSA Enc=Camellia(128) Mac=SHA256
ECDHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=Camellia(128) Mac=SHA256
DHE-RSA-CAMELLIA128-SHA256 TLSv1.2 Kx=DH Au=RSA Enc=Camellia(128) Mac=SHA256
DHE-DSS-CAMELLIA128-SHA256 TLSv1.2 Kx=DH Au=DSS Enc=Camellia(128) Mac=SHA256
```

ARIA128-GCM-SHA256	TLSv1.2	Kx=RSA	Au=RSA	Enc=ARIA128GCM(128)	Mac=AEAD
PSK-AES128-GCM-SHA256	TLSv1.2	Kx=PSK	Au=PSK	Enc=AESGCM(128)	Mac=AEAD
PSK-AES128-CCM	TLSv1.2	Kx=PSK	Au=PSK	Enc=AESCCM(128)	Mac=AEAD
PSK-ARIA128-GCM-SHA256	TLSv1.2	Kx=PSK	Au=PSK	Enc=ARIA128GCM(128)	Mac=AEAD
DHE-PSK-AES256-CCM8	TLSv1.2	Kx=DHEPSK	Au=PSK	Enc=AESCCM8(256)	Mac=AEAD
DHE-PSK-AES128-CCM8	TLSv1.2	Kx=DHEPSK	Au=PSK	Enc=AESCCM8(128)	Mac=AEAD
AES256-CCM8	TLSv1.2	Kx=RSA	Au=RSA	Enc=AESCCM8(256)	Mac=AEAD
AES128-CCM8	TLSv1.2	Kx=RSA	Au=RSA	Enc=AESCCM8(128)	Mac=AEAD
PSK-AES256-CCM8	TLSv1.2	Kx=PSK	Au=PSK	Enc=AESCCM8(256)	Mac=AEAD
PSK-AES128-CCM8	TLSv1.2	Kx=PSK	Au=PSK	Enc=AESCCM8(128)	Mac=AEAD
AES256-SHA256	TLSv1.2	Kx=RSA	Au=RSA	Enc=AES(256)	Mac=SHA256
CAMELLIA256-SHA256	TLSv1.2	Kx=RSA	Au=RSA	Enc=Camellia(256)	Mac=SHA256
AES128-SHA256	TLSv1.2	Kx=RSA	Au=RSA	Enc=AES(128)	Mac=SHA256
CAMELLIA128-SHA256	TLSv1.2	Kx=RSA	Au=RSA	Enc=Camellia(128)	Mac=SHA256
ECDSA-PSK-AES256-CBC-SHA384	TLSv1	Kx=ECDSA	Au=PSK	Enc=AES(256)	Mac=SHA384
ECDSA-PSK-AES256-CBC-SHA	TLSv1	Kx=ECDSA	Au=PSK	Enc=AES(256)	Mac=SHA1
SRP-DSS-AES-256-CBC-SHA	SSLv3	Kx=SRP	Au=DSS	Enc=AES(256)	Mac=SHA1
SRP-RSA-AES-256-CBC-SHA	SSLv3	Kx=SRP	Au=RSA	Enc=AES(256)	Mac=SHA1
SRP-AES-256-CBC-SHA	SSLv3	Kx=SRP	Au=SRP	Enc=AES(256)	Mac=SHA1
RSA-PSK-AES256-CBC-SHA384	TLSv1	Kx=RSAPSK	Au=RSA	Enc=AES(256)	Mac=SHA384
DHE-PSK-AES256-CBC-SHA384	TLSv1	Kx=DHEPSK	Au=PSK	Enc=AES(256)	Mac=SHA384
RSA-PSK-AES256-CBC-SHA	SSLv3	Kx=RSAPSK	Au=RSA	Enc=AES(256)	Mac=SHA1
DHE-PSK-AES256-CBC-SHA	SSLv3	Kx=DHEPSK	Au=PSK	Enc=AES(256)	Mac=SHA1
ECDSA-PSK-CAMELLIA256-SHA384	TLSv1	Kx=ECDSA	Au=PSK	Enc=Camellia(256)	Mac=SHA384
RSA-PSK-CAMELLIA256-SHA384	TLSv1	Kx=RSAPSK	Au=RSA	Enc=Camellia(256)	Mac=SHA384
DHE-PSK-CAMELLIA256-SHA384	TLSv1	Kx=DHEPSK	Au=PSK	Enc=Camellia(256)	Mac=SHA384
AES256-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=AES(256)	Mac=SHA1
CAMELLIA256-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=Camellia(256)	Mac=SHA1
PSK-AES256-CBC-SHA384	TLSv1	Kx=PSK	Au=PSK	Enc=AES(256)	Mac=SHA384
PSK-AES256-CBC-SHA	SSLv3	Kx=PSK	Au=PSK	Enc=AES(256)	Mac=SHA1
PSK-CAMELLIA256-SHA384	TLSv1	Kx=PSK	Au=PSK	Enc=Camellia(256)	Mac=SHA384
ECDSA-PSK-AES128-CBC-SHA256	TLSv1	Kx=ECDSA	Au=PSK	Enc=AES(128)	Mac=SHA256
ECDSA-PSK-AES128-CBC-SHA	TLSv1	Kx=ECDSA	Au=PSK	Enc=AES(128)	Mac=SHA1
SRP-DSS-AES-128-CBC-SHA	SSLv3	Kx=SRP	Au=DSS	Enc=AES(128)	Mac=SHA1
SRP-RSA-AES-128-CBC-SHA	SSLv3	Kx=SRP	Au=RSA	Enc=AES(128)	Mac=SHA1
SRP-AES-128-CBC-SHA	SSLv3	Kx=SRP	Au=SRP	Enc=AES(128)	Mac=SHA1
RSA-PSK-AES128-CBC-SHA256	TLSv1	Kx=RSAPSK	Au=RSA	Enc=AES(128)	Mac=SHA256
DHE-PSK-AES128-CBC-SHA256	TLSv1	Kx=DHEPSK	Au=PSK	Enc=AES(128)	Mac=SHA256
RSA-PSK-AES128-CBC-SHA	SSLv3	Kx=RSAPSK	Au=RSA	Enc=AES(128)	Mac=SHA1
DHE-PSK-AES128-CBC-SHA	SSLv3	Kx=DHEPSK	Au=PSK	Enc=AES(128)	Mac=SHA1
ECDSA-PSK-CAMELLIA128-SHA256	TLSv1	Kx=ECDSA	Au=PSK	Enc=Camellia(128)	Mac=SHA256
RSA-PSK-CAMELLIA128-SHA256	TLSv1	Kx=RSAPSK	Au=RSA	Enc=Camellia(128)	Mac=SHA256
DHE-PSK-CAMELLIA128-SHA256	TLSv1	Kx=DHEPSK	Au=PSK	Enc=Camellia(128)	Mac=SHA256
AES128-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=AES(128)	Mac=SHA1
SEED-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=SEED(128)	Mac=SHA1
CAMELLIA128-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=Camellia(128)	Mac=SHA1
PSK-AES128-CBC-SHA256	TLSv1	Kx=PSK	Au=PSK	Enc=AES(128)	Mac=SHA256
PSK-AES128-CBC-SHA	SSLv3	Kx=PSK	Au=PSK	Enc=AES(128)	Mac=SHA1
PSK-CAMELLIA128-SHA256	TLSv1	Kx=PSK	Au=PSK	Enc=Camellia(128)	Mac=SHA256

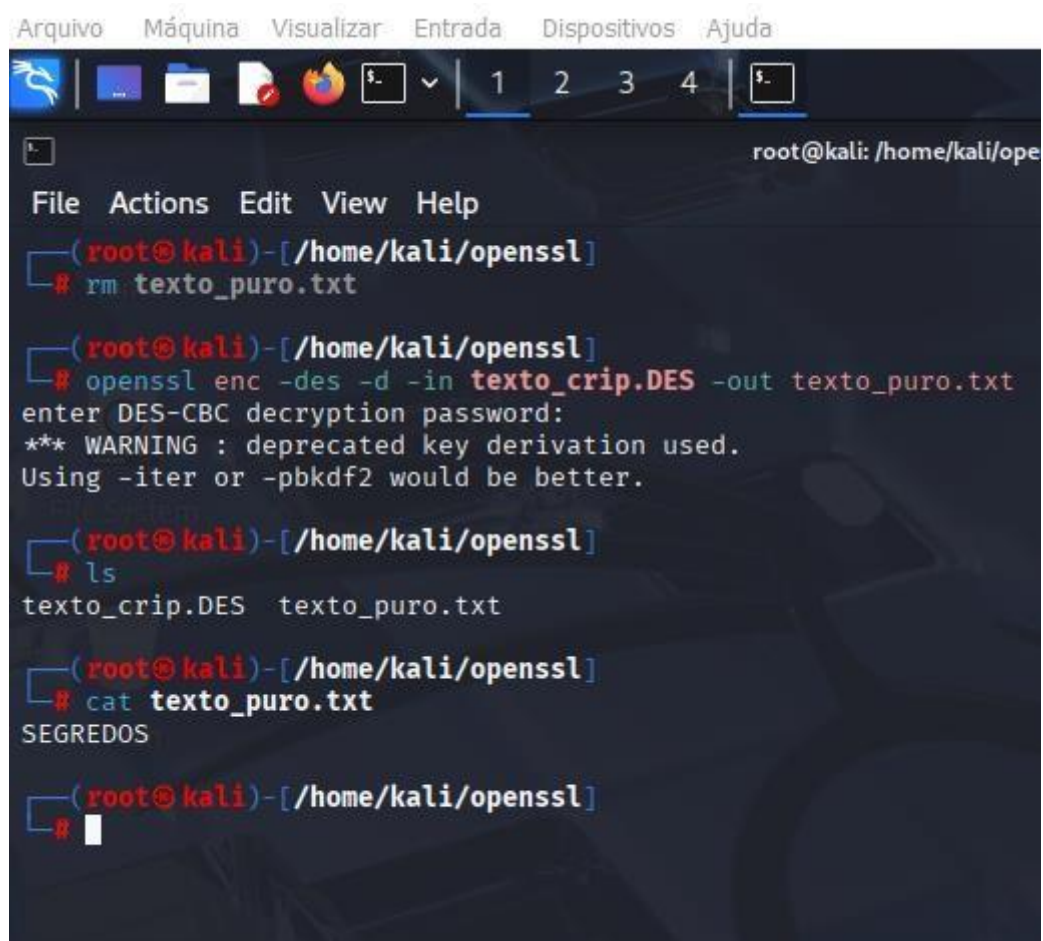
b. Escolhendo uma cifra e criptografando o arquivo (algoritmo DES):

```

root@kali: /home/kali/openssl
File Actions Edit View Help
root@kali)-[/home/kali/openssl]
# touch texto_puro.txt
root@kali)-[/home/kali/openssl]
# echo "SEGREDOS" > texto_puro.txt
root@kali)-[/home/kali/openssl]
# cat texto_puro.txt
SEGREDOS
root@kali)-[/home/kali/openssl]
# openssl enc -des -e -in texto_puro.txt -out texto_crip.DES
enter DES-CBC encryption password:
Verifying - enter DES-CBC encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@kali)-[/home/kali/openssl]
# cat texto_crip.DES
Salted__H|mmmm^##B--U*
root@kali)-[/home/kali/openssl]
# cat texto_crip.DES | xxd
00000000: 5361 6c74 6564 5f5f e699 de48 7ce5 6dac  Salted__H|.m.
00000010: b5ac 3e5e b023 1db4 4287 2d17 c355 2ab4  ..>^.#..B.-..U*

```

c. Decriptografando o arquivo (algoritmo DES):



The image shows a terminal window on a Kali Linux system. The window has a menu bar at the top with options: Arquivo, Máquina, Visualizar, Entrada, Dispositivos, and Ajuda. Below the menu bar is a toolbar with icons for file operations. The terminal prompt is root@kali: /home/kali/ope. The user enters the command rm texto\_puro.txt. Then, the user enters the command openssl enc -des -d -in texto\_crip.DES -out texto\_puro.txt. The terminal displays the message enter DES-CBC decryption password: followed by a warning: \*\*\* WARNING : deprecated key derivation used. Using -iter or -pbkdf2 would be better. The user then enters the command ls, and the terminal displays the output texto\_crip.DES texto\_puro.txt. Finally, the user enters the command cat texto\_puro.txt, and the terminal displays the output SEGREDOS.

```
File Actions Edit View Help
(root@kali)-[/home/kali/openssl]
# rm texto_puro.txt

(root@kali)-[/home/kali/openssl]
# openssl enc -des -d -in texto_crip.DES -out texto_puro.txt
enter DES-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

(root@kali)-[/home/kali/openssl]
# ls
texto_crip.DES  texto_puro.txt

(root@kali)-[/home/kali/openssl]
# cat texto_puro.txt
SEGREDOS

(root@kali)-[/home/kali/openssl]
#
```

d. Escolhendo uma cifra e criptografando o arquivo (algoritmo AES):



e. Decriptografando o arquivo (algoritmo AES):

```
(root@kali)-[/home/kali/openssl]
# rm texto_puro.txt

(root@kali)-[/home/kali/openssl]
# openssl enc -aes-256-cbc -d -in texto_crip.AES -out texto_puro.txt
enter AES-256-CBC decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

(root@kali)-[/home/kali/openssl]
# ls
texto_crip.AES  texto_puro.txt

(root@kali)-[/home/kali/openssl]
# cat texto_puro.txt
SEGREDOS

(root@kali)-[/home/kali/openssl]
# cat texto_puro.txt | xxd
00000000: 5345 4752 4544 4f53 0a          SEGREDOS.
```

1. Atividade (printar as telas para comprovação da atividade):

e.1.1. Crie um arquivo nome.txt usando editor de texto com uma frase de sua escolha. Em seguida criptografar esse arquivo com os algoritmos DES e AES. Demonstrar os resultados.

1. Quando se criptografa o mesmo arquivo com o DES ou AES duas vezes, por que o resultado é diferente com o mesmo valor de entrada? Justifique. (Dica: usar o cat

arquivo\_criptografado | xxd)

R: Quando você criptografa o mesmo arquivo duas vezes usando AES ou DES, mesmo com a mesma chave e os mesmos dados, o resultado é diferente porque a criptografia moderna usa dois mecanismos importantes para aumentar a segurança: o salt e o IV (Vetor de Inicialização).

O salt é um valor aleatório adicionado antes da criptografia para evitar que duas entradas iguais gerem saídas idênticas. Ele é comum em sistemas que usam senhas, como quando o

OpenSSL é executado com a opção salt. Se um salt for usado, mesmo a mesma senha produzirá uma chave diferente a cada vez, alterando o resultado final.

Já o IV (Vetor de Inicialização) é usado em modos de operação como CBC ou CTR para garantir que blocos de dados idênticos não gerem cifras repetidas. Ele funciona como um "valor inicial aleatório" que é misturado com os dados antes da criptografia. Assim, mesmo que o arquivo e a chave sejam os mesmos, um IV diferente fará com que a saída seja completamente distinta.

Por isso, ao criptografar o mesmo arquivo duas vezes, o salt (se aplicável) e o IV (sempre presente em modos seguros como CBC) garantem que o resultado mude, evitando padrões que poderiam ser explorados por ataques. Essa aleatoriedade controlada é essencial para manter a segurança da criptografia.

## 2. Compare os algoritmos DES e AES. Qual é o algoritmo mais vulnerável. Justifique.

R: O DES (Data Encryption Standard) é mais vulnerável que o AES (Advanced Encryption Standard) porque usa uma chave muito pequena (apenas 56 bits), o que o torna fraco contra ataques de força bruta com hardware moderno. Além disso, o DES opera com blocos de 64 bits, menos eficientes que os 128 bits do AES. Já o AES suporta chaves de 128, 192 ou 256 bits, resistindo melhor a ataques criptográficos e sendo o padrão atual para segurança robusta. Enquanto o DES já foi quebrado na prática, o AES ainda é considerado seguro quando configurado corretamente.

## Parte 2: Criptografia Assimétrica usando o OpenSSL (Algoritmos RSA)

a. Destinatário gerando as chaves (Criando a chave privada RSA):

b. Destinatário gerando as chaves (Criando a chave pública RSA):

```
(root@kali)~/home/kali
# openssl rsa -in c_priv -pubout > c_pub
Enter pass phrase for c_priv:
writing RSA key

(root@kali)~/home/kali
# cat c_pub
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAnFmXf2LtArIT7PQCEb
qphThr5UsRTZzj4+8XIc/cC+L92SmOtkcM/20C/o44yctMOXcTmLIIRmTyRv8w57
R/EEZvpZV71Skwkg3MR1BIZ56bR0Y9Q57XtA2y8Z7QgtQdm0ALP1mMfEcRpEPoiS
6Fch0tZAGLLXKxIkIynG/ygP8m4JufmEKyPsDzU08wHPsBCNtunhNxSsGj1l/T8+
ErriWbLw4bKzdMp0sdJwpgP+ju10V+MI4cUa1Vkk18noYC10rYfnKYFzzYIdr9qw
X4vLElwE0wQSBn7OWHXVgIdcNihWSxFm10WLnS00TwtRUSconfvwVSx8WRgVgP8+
ewIDAQAB
-----END PUBLIC KEY-----
```

c. Remetente cria o arquivo:

```
(root@kali)~/home/kali
# echo "CRIPTOGRAFIA CHAVE PUBLICA" > teste.txt

(root@kali)~/home/kali
# cat teste.txt
CRIPTOGRAFIA CHAVE PUBLICA
```

d. Remetente criptografando com a chave pública RSA do destinatário:

```
(root@kali)~/home/kali
# openssl rsautl -in teste.txt -encrypt -pubin -inkey c_pub > teste.rsa
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.

(root@kali)~/home/kali
# cat teste.rsa
+E]wB+Δ+T\*!0BD+++6++0+++ x+P++Xs!f--+4+-W[*****uls*X,l+{8*(+S<+>+%/+****i*****Dn+5++3*+>+ã+$ZW6+****^$.****t(f+<+)+.+
s*****{
3*****0+;*KwZ9W+}*****x++H+++M++++'+.+*Tv+Fx+
>KdCPME5+v+++,*g+'%+++0++++7MM5 *|.+'*0j<+*
```

e. Destinatário decryptografando com a sua chave privada RSA:

```
(root@kali)~/home/kali
# openssl rsautl -in teste.rsa -decrypt -inkey c_priv > teste.rec
The command rsautl was deprecated in version 3.0. Use 'pkeyutl' instead.
Enter pass phrase for c_priv:

(root@kali)~/home/kali
# cat teste.rec
CRIPTOGRAFIA CHAVE PUBLICA
```

2. Atividade (printar as telas para comprovação da atividade):

2.1. Quem deve gerar o par de chaves? O destinatário ou o remetente? Justifique.

R: Em criptografia RSA, o destinatário (quem vai receber a mensagem) **gera o par de chaves (pública e privada)**. Ele então **distribui sua chave pública** para que qualquer remetente possa criptografar mensagens para ele. A chave privada **nunca é compartilhada** e serve apenas para o destinatário decifrar a mensagem recebida.

2.2. Criptografar o arquivo (criar arquivo) e enviar para o outro componente do grupo.

Quem criptografa o arquivo? O remetente ou o destinatário? Justifique.

R: O **remetente criptografa o arquivo** antes de enviá-lo, para garantir que apenas o destinatário possa abri-lo. Ele usará a chave pública do destinatário para isso. Assim, mesmo que alguém intercepte o arquivo, não poderá lê-lo.

2.3. Qual chave que é usada para criptografar o arquivo? Justifique.

R: A **chave pública** do destinatário é usada para criptografar, pois ela é segura para ser compartilhada. Apenas a **chave privada correspondente**, que só o destinatário possui, pode decryptografar esse conteúdo.

2.4. Decryptografar o arquivo. Quem decryptografa o arquivo? O remetente ou o destinatário? Justifique.

R: A mensagem foi criptografada para o destinatário. Assim, **apenas ele pode decryptografar**, pois só ele tem a **chave privada** correspondente à chave pública usada na criptografia.

2.5. Qual chave que é usada para decryptografar o arquivo? Justifique.

R: A **chave privada** é usada para decryptografar porque é a única que pode desfazer a criptografia feita com a chave pública correspondente. E como essa chave é **secreta e exclusiva do destinatário**, garante a confidencialidade.

### Parte 3: Gerando o Hash do arquivo usando o OpenSSL e Assinando um arquivo usando o OpenSSL

a. Gerando hashes (MD5, SHA1, SHA256, SHA512):



```
(root@kali)-[/home/kali]
# echo "ALGORITMO DE HASH" > teste.txt

(root@kali)-[/home/kali]
# cat teste.txt
ALGORITMO DE HASH

(root@kali)-[/home/kali]
# openssl dgst -sha256 teste.txt
SHA2-256(teste.txt)= fca04653ae90c51ba6f5310d2bd2702f4832a855b261d50ce15af9d9dfa18b52

(root@kali)-[/home/kali]
#
```

b. Remetente gerando as chaves (Criando a chave privada RSA):



```

(root@kali)-[/home/kali]
# openssl dgst -sha256 teste.txt
SHA2-256(teste.txt)= fca04653ae90c51ba6f5310d2bd2702f4832a855b261d50ce15af9d9dfa18b52

(root@kali)-[/home/kali]
# openssl genrsa -aes-256-cbc 2048 > c_priv
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

(root@kali)-[/home/kali]
# cat c_priv
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFNTBfBgkqhkiG9w0BBQ0wUjAxBgkqhkiG9w0BBQwwJAQYyJLe4wa318DjEHXS
dD+7QwICCAAwDAYIKoZIHvcNAgkFADAdBgIghkgBZQMEASoEEM1yCUH9YE9cE6Yi
OEa694gEgTQ0eMXUxkcLgIVNPQgLPkbM05X5UvFZ9LIiSEEWQ+atBaQ5DCPPxwC
z1d3GF4JLk+Z1FFKfCc+ZepaU4eEgALg15L9FCbjYqAQo/cHoP8Yozc+GmUeQxo+
HKefoto0noGEdKxtEXo6Au4gKLVhDjgZsmlZa43VmoEKEF41cE3CiUNsxFTgR0
BUQxo7B/Tr1bXoppQx/z57/xrcbh2XcQZrtCM5ILYDca1KmwgF70mGY32wgFJumb
NmzAXmDc00er48AG+lQ0f3IMJ/H2gS6or+U2hCEkn3rCry9YHoXLf0wSAn3k79YT
B3MqibQuQF70E6gE8+81dxNLqKopAPSjyplQvxE333LAPxrqHS6vYvoFee5EAFZ
3BAeUaIVJXWE0Qip3SHFvg7br5rGb26816HcA3Mol1oc1EH1V+5c8ZiGHGswDyNh
82dWmbEEZULuBNUpENf5i/3yDQhN4YBdCqT8IZ9n1Tu0n6XAKJWjf3YG2FSvqU5R
ISQhszeMGch3dR/4je12GsdarPi+5Df6f/S867ttSmK/+qqq/zPMREWauvoJF/v
sIBQKr0pNrl8GoqUHa5XIAzjt6fUAHA3EGrVbwZfiw+p9/YNOp16Kj0gAYkXV
5gdenS13JmTYMuZZJULwjB8C5qnzSjL03d03cT9NxGnsNaCKqscWYa8zLjdPvZeE
E0ga7eEcVwu74yCYrKQM83HB6/qxDNQGtGzclZOKOc04tyIfYKSnXt/683y+M72U
OkIL58LE402BAGU8aJ0jzSLBnLNx+brIcb+pNRzKgs11iAaG0LVQLw8+xEJdc
MahGlorDZ1QgBabIfv7GGOP1KsIH3+nTA+5P1w28MM82pSRsk6BqJGhpeAHy9mNH
6FC7beWD29Np7PKxHMKRP+uCWxAQF5P8Wav0+TxAz4I8dv91gIK7J43eQrGzpP0
SCOR4+1Vgr4yLkWMZ5VGNnIdzFNyk2qDTAu0BwLBUafDCE+g4mLQA82CkGEZd3n8
0j0IcnreNNYXFuFOTbdrQOnxEKGoakbXCwT2gDG7naF/kPgDJmmoZCTFQKRLAbc
9Z50/CX9GexblcNZFG8yHIhDvI5ROUE6djYHBMZR2apbEpQc4K1K58bPSmxcTb9bN
8xYIN/WB99Hs0f9VIEJXK6CXG0teseEv/n06LrV68fd1km+vjqgCyx31Y07+dzh5
1oB7JaKhKHJMy6gwxjR904Lmhv0WbZIM4XifKtu89IElr0jKRN2+SfuTxv+b8Qf
oh873W7UG0ro6LLABOGL6BsNOVV1Z/ciFZgG5dwT7L+7U4J1dxrj2nu46XNQtsmk
Rin8vUPI6feJjZV7cDoudly3J/VL+d3MypqTW6KjJLmi+SIkW6gYAT7edY2Ru045
JwJbEFyD8aJs0+3EoggIOKPL4YIY/Crpl1ebCEEFZ/8vcTbEK/M43bzcMDaQuKC
u6noAbjQV3ANvraUTYUWOUcJk4eUQYcEk5J59Ly2vkAM3BXpHcWj/SzwJFthJl04
yR1K/YxGZ3/CfaCqzLiN0X0ow7HqqGd26yI36PS+VkkwRqXSUUXhYAcgCZTT+nP
otsSTL58XA06cZqi+c7B60z8gGTitt0Pw7Fk871X0LhkiootlnUe7V4=
-----END ENCRYPTED PRIVATE KEY-----

```

Remetente gerando as chaves (Criando a chave pública RSA):

```

(root@kali)-[/home/kali]
# openssl rsa -in c_priv -pubout > c_pub
Enter pass phrase for c_priv:
writing RSA key

(root@kali)-[/home/kali]
# cat c_pub
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAttRcLasffQb3*2tAfizR
zUrAMLhb8F/hw3S+V5vLK3GDtDLP6DFMaTjji/uQzK8p7xOQ3KsUpD+qp0ZTIec8
azQCycelmlC65wC7OZRPLmqkor5LrAu5MytTx6IoI7G5eo3W2wQysUsnuOKLCh4T
usWF3eZXR0SAT7z47/6oXuTuof5qteCzQ1vCmIeXN3sTcZLP7wMaCsWeg47E42Sk
UR1k3FPLGQRaj8Qbc/zER/jFnQpr5GZVUV7sPjYm9IqSeMQirW2W6vqgGhyYB+P4
0bMXQ5lvWDgqguS02Ck0izT2vEWm61evWy2JuJE1XJKoBd0SeEE/0y1hr/bPCyhy
ZQIDAQAB
-----END PUBLIC KEY-----

```

c. Remetente cria o arquivo:

```
(root@kali)-[/home/kali]
# echo "ASSINATURA DIGITAL" > teste.txt

(root@kali)-[/home/kali]
# cat teste.txt
ASSINATURA DIGITAL
```

d. Remetente assina com a chave privada RSA dele:

```
(root@kali)-[/home/kali]
# openssl dgst -sha256 -sign c_priv -out teste.assinado.sha256 teste.txt
Enter pass phrase for c_priv:

(root@kali)-[/home/kali]
# cat teste.assinado.sha256 | xxd
00000000: b39b c718 8a7d 6a51 26de 7203 6ef9 dfb4  ....}jQ6.r.n...
00000010: 81cc ca92 f681 5aa2 e9e1 0819 d88b f68d  ....Z.....
00000020: 2a34 29f8 fd3c 31d9 54ba a0c6 735c 18d8  *4) <1.T...s\..
00000030: ab6f b9f6 548e 09b2 0d37 5266 8600 6f8a  .o..T...7Rf..o.
00000040: 7f1e 0a45 48b8 cd1d 8071 5137 4cb6 11a6  ..EH...qQ7L...
00000050: 8b49 a108 9535 a5fe cba8 7414 06f3 8e75  .I...5....t...u
00000060: 1e83 112b df79 e226 2e42 0fb2 b802 2b40  ...+..y.G.B...+@
00000070: 6062 a627 9af3 caf5 6822 55e6 e0e2 028a  `b.'...h"U....
00000080: 1954 3cfc 5215 3db3 99bd 0312 ed8f bb2c  .T< R.=.....,
00000090: 168a 6e03 8028 7798 2dc8 c176 a2fb 72cf  ..n..(w.-..v..r.
000000a0: bc3b e032 2245 a4d8 6b76 f2df 5446 3372  .; 2"E..kv..TF3r
000000b0: 3e03 d12b 4b8a 3c40 2c07 f775 86c7 120c  >..+K.<@, ..u...
000000c0: 732a c117 7fe4 ddc9 2d47 8088 0d29 8725  s*.....-G...).%
000000d0: 9aab f40f 509b 4bc7 8b34 d759 d55d 61f1  ....P.K..4.Y.Ja.
000000e0: 9e0d 2189 d958 af2a aa49 7fa9 dab9 bd33  ..!..X.*.I....3
000000f0: 203a 6f10 e3ec 358d 79d8 464e dfd7 c4b1  :o...5.y.FN....
```

e. Destinatário verifica a assinatura com a chave pública do remetente:

```
(root@kali)-[/home/kali]
# openssl dgst -sha256 -verify c_pub -signature teste.assinado.sha256 teste.txt
Verified OK
```

3. Atividade (printar as telas para comprovação da atividade):

3.1. Quem deve gerar o par de chaves? O destinatário ou o remetente? Justifique.

R: Em assinaturas digitais (que utilizam funções de **hash + criptografia assimétrica**), o remetente gera o par de chaves (pública e privada).

Ele **assina o arquivo com sua chave privada**, e o destinatário **verifica a assinatura com a chave pública do remetente**.

3.2. Assinar o arquivo (criar arquivo) e enviar para o outro componente do grupo. Quem assina o arquivo? O remetente ou o destinatário? Justifique.

R: O objetivo da assinatura digital é garantir que a mensagem realmente veio do remetente e não foi alterada. Por isso, **o remetente assina o arquivo**, criando um **hash** e criptografando esse hash com sua **chave privada**.

3.3. Qual chave que é usada para assinar o arquivo? Justifique.

R: A **chave privada do remetente** é usada para assinar o hash do arquivo. Isso assegura que só ele poderia ter feito aquela assinatura, pois só ele possui essa chave.

3.4. Verificar a assinatura do arquivo. Quem verifica a assinatura do arquivo? O remetente ou o destinatário? Justifique.

R: O **destinatário** recebe o arquivo e a assinatura. Ele usa a **chave pública do remetente** para verificar se a assinatura é válida, ou seja, se o hash assinado confere com o conteúdo real do arquivo.

3.5. Como o destinatário consegue identificar a autenticidade do usuário remetente e a integridade do arquivo? Justifique.

R:

**Autenticidade**

**do**

**remetente:** O destinatário usa a

**chave pública do remetente** para descriptografar a assinatura. Se a operação funcionar, significa que a assinatura **só pode ter sido feita com a chave privada correspondente**, provando que foi o remetente quem assinou.

**Integridade do arquivo:**

O destinatário **gera o hash do arquivo recebido** e compara com o **hash obtido da assinatura**.

Se os dois hashes forem iguais, o conteúdo **não foi alterado** — está íntegro.

