OPENSSL-REPORT

GROUP-10

BINOY KRISHNA PAL <CS24MTECH11009>
ISHNOOR JAIN <CS24MTECH14021>

Part A: Authentication Using RSA

Alice:

Step 1:

Firstly we will make alice_passphrase.txt which store the secure passphrase which protect the private key on each system. This ensures that even if someone gains access to the file, they cannot use the private key without knowing the passphrase. It is done as the question states, **"The private key on each system must be protected with a secure passphrase,"**

Command used: echo "your_secure_passphrase" > alice_passphrase.txt

Result: alice_passphrase.txt will be created which stores our passphrase "your_secure_passphrase"

Step 2:

We need to create alice's private key

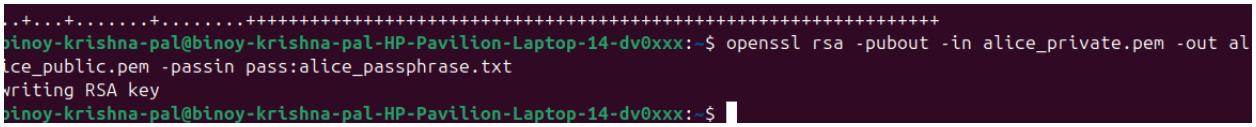
Command used: openssl genpkey -algorithm RSA -out alice_private.pem -aes256 -pass pass:alice_passphrase.txt -pkeyopt rsa_keygen_bits:2048

Result: Create 2048 bit private key for alice

Step 3:

Now we have to extract public key from the private key generated

Command used:openssl rsa -pubout -in alice_private.pem -out alice_public.pem -passin pass:alice_passphrase.txt



Step 4:

Now we will create file named alice_roll.txt which stores my roll number as the data inside it.

Command used: echo ”cs24mtech11009” >bob_roll.txt

Step 5:

After creating a text file with their roll number we have to sign it using the private key.Generate a signature using his private key

Command used: openssl dgst -sha256 -sign alice_private.pem -passin pass:alice_passphrase.txt -out alice_signature.bin alice_roll.txt

Result:

```
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/opens
sl_assignment$ echo "cs24mtech11009" > alice_roll.txt
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/opens
sl_assignment$ openssl dgst -sha256 -sign alice_private.pem -passin pass:alice_p
assphrase.txt -out alice_signature.bin alice_roll.txt
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/opens
```

Step 6:

I share to bob alice_public.pem, alice_roll.txt, alice_signature.bin.

Step 7:

Last step is signature verification. I(alice) will verify the received signature(bob signature) using the bob's public key.

Command used: openssl dgst -sha256 -verify bob_public.pem -signature bob_signature.bin bob_roll.txt

```
:
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl dgst -sha256 -verify bob_public.pem -signature bob_signature.bin bob_roll.txt
Verified OK
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
```

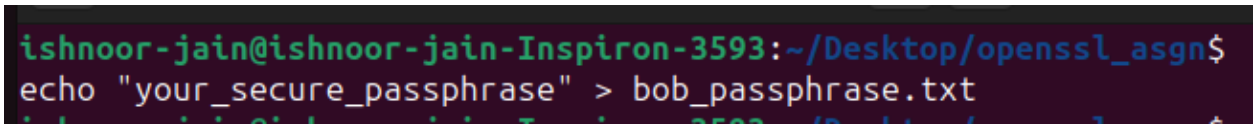
Bob:

Step 1:

Firstly we will make bob_passphrase.txt which store the secure passphrase which protect the private key on each system. This ensures that even if someone gains access to the file, they cannot use the private key without knowing the passphrase. It is done as the question states, **"The private key on each system must be protected with a secure passphrase,"**

Command used: echo "your_secure_passphrase" > bob_passphrase.txt

Result: bob_passphrase.txt will be created which stores our passphrase “your_secure_passphrase"

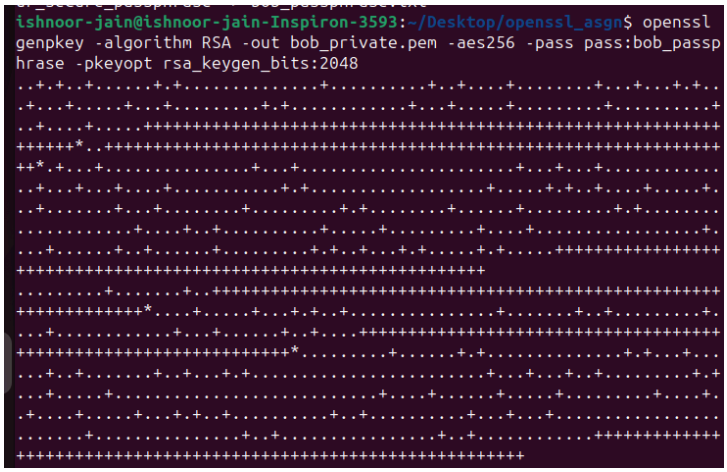


Step 2:

We need to create bob’s private key

Command used: openssl genpkey -algorithm RSA -out bob_private.pem -aes256 -pass pass:bob_passphrase -pkeyopt rsa_keygen_bits:2048

Result: Create 2048 bit private key for bob

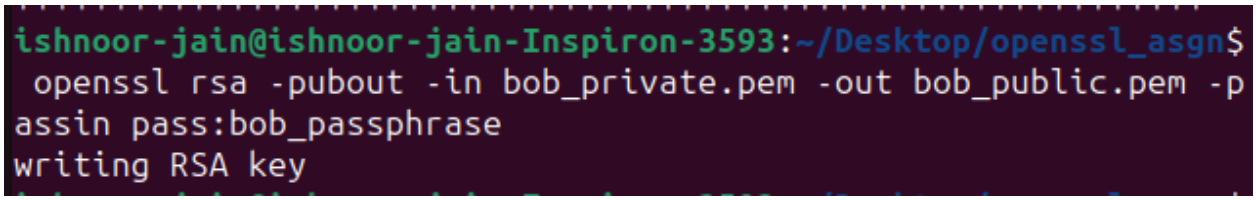


Step 3:

Now we have to extract public key from the private key generated

Command used:openssl rsa -pubout -in bob_private.pem -out bob_public.pem -passin pass:bob_passphrase

Result:

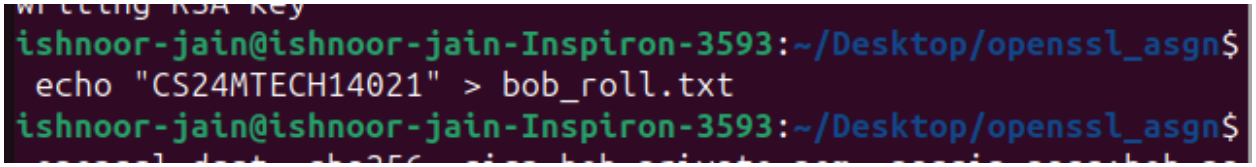


Step 4:

Now we will create file named bob_roll.txt which stores my roll number as the data inside it.

Command used: echo ”CS24MTECH14021” >bob_roll.txt

Result:

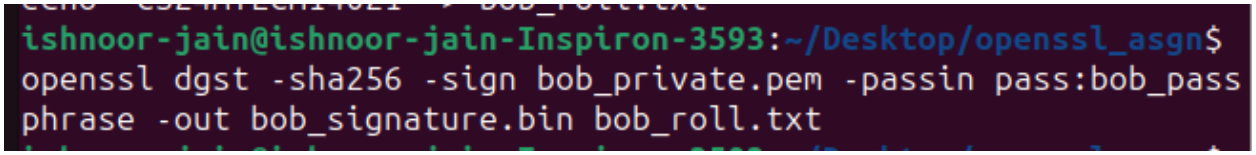


Step 5:

After creating a text file with their roll number we have to sign it using the private key.Generate a signature using his private key

Command used: openssl dgst -sha256 -sign bob_private.pem -passin pass:bob_passphrase -out bob_signature.bin bob_roll.txt

Result:



Step 6:

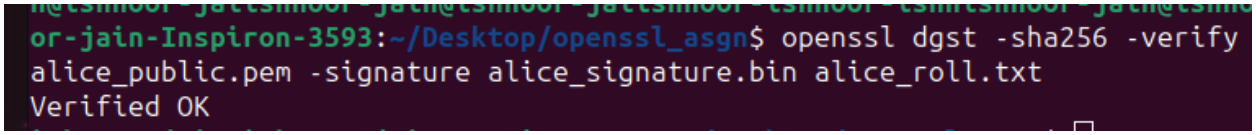
I share to alice bob_public.pem, bob_roll.txt, bob_signature.bin.

Step 7:

Last step is signature verification. Each participant will verify the received signature using the sender's public key.

Command used: openssl dgst -sha256 -verify alice_public.pem -signature alice_signature.bin alice_roll.txt

Result:



Terminal showing all steps:

Part B: Key Exchange Using Diffie-Hellman (DFH)

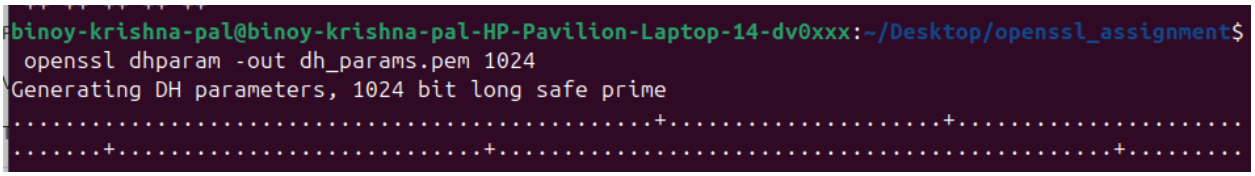
Alice:

Step 1: Generate Diffie-Hellman Parameters

Alice generates the DFH parameters using OpenSSL:

Command used:openssl dhparam -out dh_params.pem 1024

Screenshot:



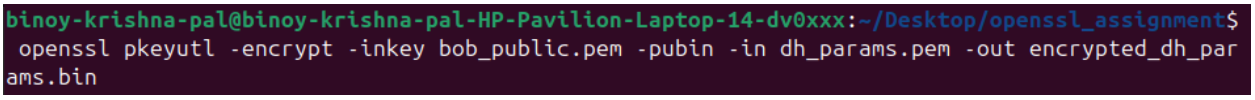
Step 2: Encrypt Parameters with Bob's RSA Public Key

Alice encrypts the DFH parameters using Bob's public key to ensure secure transmission:

Command used: openssl pkeyutl -encrypt -inkey bob_public.pem -pubin -in dhparams.pem -out encrypted_dh_params.pem

Alice securely sends `encrypted_dh_params.bin` to Bob.

Screenshot:



Step 3: Generate Diffie-Hellman Key Pair

Once Bob has confirmed the integrity of the DFH parameters, Alice generates her own DH key pair.

Command used: openssl genpkey -paramfile dh_params.pem -out alice_DH_private.pem

openssl pkey -in alice_DH_private.pem -pubout -out alice_DH_public.pem

```
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl genpkey -paramfile dh_params.pem -out alice_DH_private.pem
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
```

```
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl pkey -in alice_DH_private.pem -pubout -out alice_DH_public.pem
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
```

Step 4:

Now both, Alice and Bob have exchanged their DFH public keys to compute a shared secret.

Step 5: Compute Shared Secret

After receiving Bob's DH public key, Alice computes the shared secret.

Command used: openssl pkeyutl -derive -inkey alice_DH_private.pem -peerkey bob_DH_public.pem -out alice_shared_secret.bin

```
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl pkey -in alice_DH_private.pem -pubout -out alice_DH_public.pem
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl pkeyutl -derive -inkey alice_DH_private.pem -peerkey bob_DH_public.pem -out alice_sha
red_secret.bin
```

Step 6: Derive AES Key

Alice derives an AES key from the shared secret.

Command used: openssl dgst -sha256 -binary alice_shared_secret.bin | head -c 32 > AES_key.bin

```
 openssl pkeyutl -derive -inkey alice_DH_private.pem -peerkey bob_DH_public.pem -out alice_sha
red_secret.bin
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl dgst -sha256 -binary alice_shared_secret.bin | head -c 32 > AES_key.bin
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
```

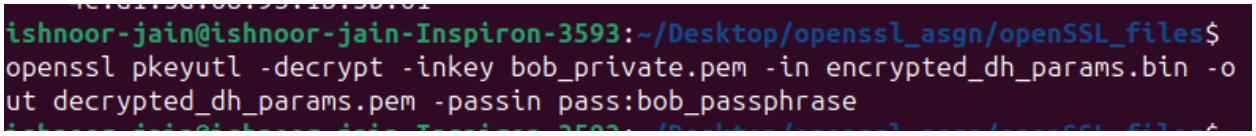
Bob:

Step 1:

After Alice has generate the Diffie-Hellman (DFH) parameters and encrypted using Bob's RSA public key and shared securely I (Bob) have decrypted the received DFH parameters using my private RSA key.

Command: openssl pkeyutl -decrypt -inkey bob_private.pem -in encrypted_dh_params.bin -out decrypted_dh_params.pem -passin pass:bob_passphrase

Result:



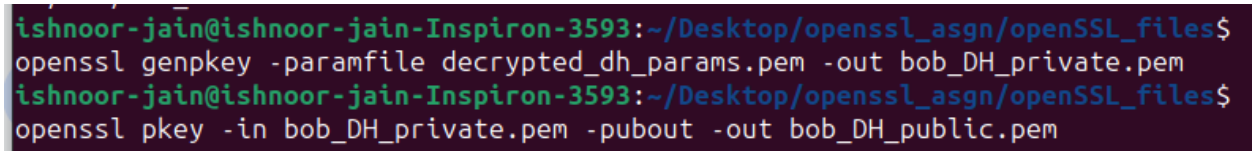
Step 2:

Now Both participants will independently generate DFH key pairs based on the shared parameters.

Command:

openssl genpkey -paramfile decrypted_dh_params.pem -out bob_DH_private.pem
openssl pkey -in bob_DH_private.pem -pubout -out bob_DH_public.pem

Result:

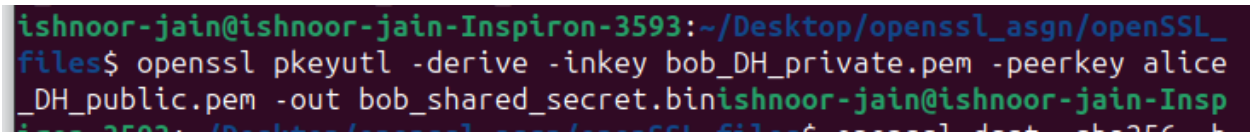


Step 3:

Alice and Bob will exchange their DFH public keys files bob_DH_public.pem and compute a shared secret.

Command:openssl pkeyutl -derive -inkey bob_DH_private.pem -peerkey alice_DH_public.pem -out bob_shared_secret.bin

Result:



Step 4:

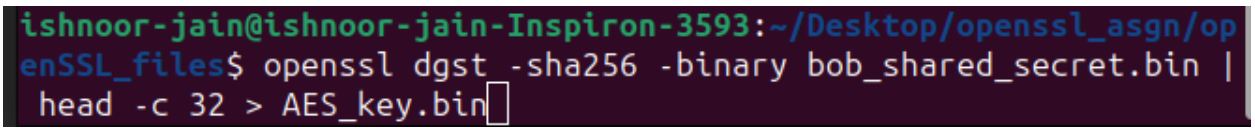
Now both, Alice and Bob have exchanged their DFH public keys to compute a shared secret.

Step 5:

The DFH shared secret will be used to derive a symmetric AES key.

Command:openssl dgst -sha256 -binary bob_shared_secret.bin | head -c 32 > AES_key.bin

Result:



Part C: Secure File Sharing Using AES

Alice:

1. File Encryption

To securely send a file (`text.txt`) to Bob, I encrypted it using AES-256-CBC and send it to Bob

Command: openssl enc -aes-256-cbc -salt -pbkdf2 -in text.txt -out encrypted_text.bin -pass file:AES_key.bin

```
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 echo "hi, i am binoy" > text.txt
binoy-krishna-pal@binoy-krishna-pal-HP-Pavilion-Laptop-14-dv0xxx:~/Desktop/openssl_assignment$
 openssl enc -aes-256-cbc -salt -pbkdf2 -in text.txt -out encrypted_text.bin -pass file:AES_ke
y.bin
```

Bob:

2. File Decryption

After receiving the encrypted file from Alice Bob have decrypted the file using the shared AES key to retrieve the original content.

Command: openssl enc -d -aes-256-cbc -pbkdf2 -in encrypted_secret.bin -out decrypted_secret.txt -pass file:AES_key.bin

```
ishnoor-jain@ishnoor-jain-Inspiron-3593:~/Desktop/openssl_asgn/openS
SL_files$ openssl enc -d -aes-256-cbc -pbkdf2 -in encrypted_secret.b
in -out decrypted_secret.txt -pass file:AES_keyishnoor-jain@ishnoor-
ishnoor-jain@ishnoor-jain-Inspiron-3593:~/Desktop/openssl_asgn/openS
SL_files$
```

Anti-Plag Statement

We certify that this assignment/report is the result of our collaborative work, based on our collective study and research. All sources, including books, articles, software, datasets, reports, and communications, have been properly acknowledged. This work has not been previously submitted for assessment in any other course unless specific permission was granted by all involved instructors.

We also acknowledge the use of AI tools, such as LLMs (e.g., ChatGPT), for assistance in refining this assignment, if used. We have ensured that their usage complies with the academic integrity policies of this course. We pledge to uphold the principles of honesty, integrity, and responsibility at CSE@IITH.

Additionally, we understand our duty to report any violations of academic integrity by others if we become aware of them.

Names <Roll Nos>:BINOY KRISHNA PAL<CS24MTECH11009>

:ISHNOOR JAIN<CS24MTECH14021>

Date:23/02/2025

Signatures: BKP

: IS