Cryptography Basics Assignment

1. Introduction to Cryptography

Cryptography is the practice of securing information through encoding methods to ensure confidentiality, integrity, and authenticity. The key concepts covered in this assignment are encryption, hashing, and digital signatures.

2. Types of Encryption

Symmetric Encryption (AES-256)

- Uses a single key for both encryption and decryption.
- Example: Advanced Encryption Standard (AES-256) is widely used for secure data transmission.

Asymmetric Encryption (RSA)

- Uses a pair of keys: Public Key (encryption) and Private Key (decryption).
  - Example: RSA (Rivest-Shamir-Adleman) is commonly used for secure key exchanges.

3. Hashing Algorithms

MD5 (Message Digest Algorithm 5)

- Produces a 128-bit hash value.
- Considered weak due to vulnerabilities to collision attacks.

SHA-256 (Secure Hash Algorithm 256-bit)

- Produces a 256-bit hash value.
  - Stronger than MD5, widely used in blockchain and digital signatures.

4. Tasks

Task 1: AES-256 Encryption & Decryption using OpenSSL

Step 1: Create a sample text file:

echo "This is a plain Text" > plaintext.txt

Step 2: Encrypt the file using AES-256:

openssl enc -aes-256-cbc -salt -in plaintext.txt -out encrypted.bin -pass pass:yourpassword

```
❯ openssl enc -aes-256-cbc -salt -in plaintext.txt -out encrypted.bin -pass pass:meraaz
```

Step 3: Verify encrypted file:

```
❯ hexdump -C encrypted.bin | head
00000000  53 61 6c 74 65 64 5f 5f  31 bb 09 81 14 4d 93 7f  |Salted__1....M..|
00000010  28 d3 50 04 91 76 38 d4  75 ae e8 b7 7a 7a 16 84  |(.P..v8.u...zz..|
00000020  06 54 b8 3a a3 b5 1c 46  7a 28 fe f7 74 a5 09 61  |.T.:...Fz(..t..a|
00000030  f2 79 1c 24 ae 81 ed 08  7b bb c7 33 2c ea 5b 70  |.y.$....{..3,.[p|
00000040  c1 0e ad 59 e3 0a a1 ea  b8 e7 b3 ac d6 37 a1 ea  |...Y.........7..|
00000050
```

hexdump -C encrypted.bin | head

Step 4: Decrypt the file:

openssl enc -aes-256-cbc -d -in encrypted.bin -out decrypted.txt -pass pass:yourpassword

```
❯ openssl enc -aes-256-cbc -d -in encrypted.bin -out decrypted.txt -pass pass:meraaz
```

Step 5: Compare original and decrypted files:

diff plaintext.txt decrypted.txt

```
❯ diff plaintext.txt decrypted.txt
```

No output means no changes.

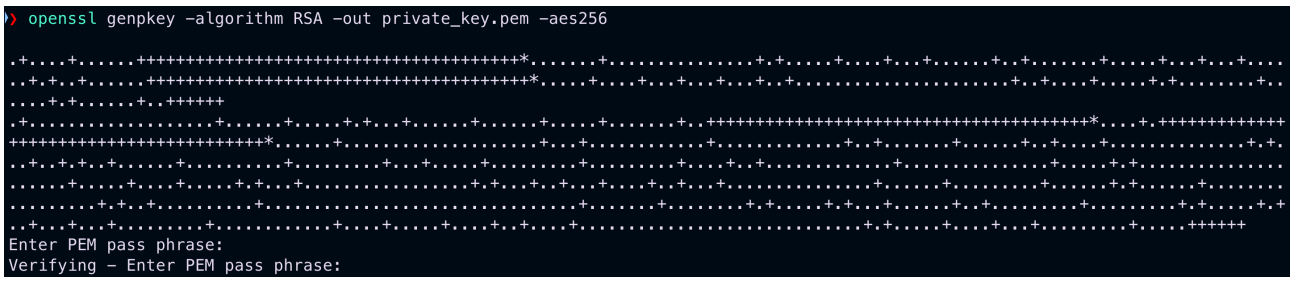
```
❯ diff plaintext.txt decrypted.txt
2c2
< I am just testing the basics of hashing.
\ No newline at end of file
---
> I am just testing the basics of hashing.....
\ No newline at end of file
❯
```

And if the texts are different this error will be shown

Task 2: RSA Key Pair Generation using OpenSSL

Step 1: Generate a private key:

openssl genpkey -algorithm RSA -out private_key.pem -aes256



`openssl genpkey`

- Generates a private key.

`-algorithm RSA`

- Specifies that the key should use the RSA algorithm.

`-out private_key.pem`

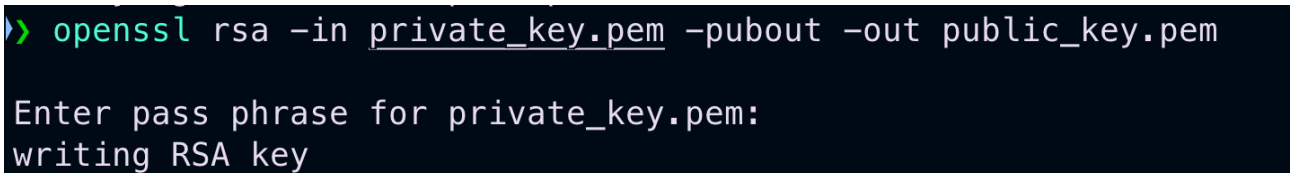
- Saves the generated private key to a file named `private_key.pem`.

`-aes256`

- Encrypts the private key using AES-256 encryption.
- You'll be prompted to enter a passphrase to protect the key.

Step 2: Extract the public key:

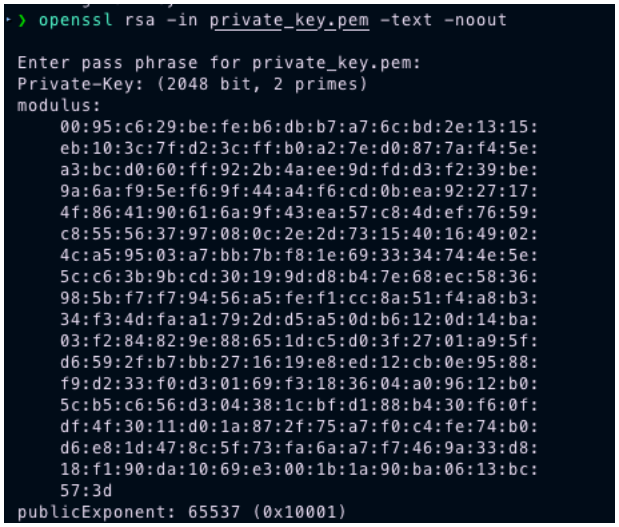
openssl rsa -in private_key.pem -pubout -out public_key.pem



Step 3: Check key details:

openssl rsa -in private_key.pem -text -noout

Note: here I used meraaz as a common password.



Task 3: Verify Hash Integrity of a File using SHA-256

Step 1: Generate SHA-256 hash:

openssl dgst -sha256 plaintext.txt

```
❯ openssl dgst -sha256 plaintext.txt

SHA2-256(plaintext.txt)= 9b3ba1ec570892874e1072a8c6505e492a91ad7a440316b1209bf0be4ab4ff81
```

`openssl dgst`

- Calls the OpenSSL **digest** command, which computes message digests (hashes).

`-sha256`

- Specifies that the **SHA-256** algorithm should be used for hashing.

`plaintext.txt`

- The input file whose hash is being computed.

Step 2: Save hash output to a file:

openssl dgst -sha256 plaintext.txt > hash.txt

Step 3: Verify hash integrity by recomputing and comparing:

openssl dgst -sha256 -c plaintext.txt

5. Conclusion

This assignment covered fundamental cryptography concepts, including AES-256 encryption, RSA key pair generation, and SHA-256 hashing. These methods are crucial in ensuring secure communication and data integrity.

Appendix: OpenSSL Installation

If OpenSSL is not installed, use the following command to install it:

```
sudo apt install openssl  # For Linux

brew install openssl  # For macOS
```