Task 1 – AES Encryption

Task 1A: Encrypt a file using AES-128-CBC

Let's create a file named secret.txt

Encrypt the file with AES-128-CBC pass:mypassword

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task A>echo "This file contains top secret information." > secret.txt

C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task A>openssl enc -aes-128-cbc -salt -in secret.txt -out secret.enc -pass pass:mypassword
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Task 2

Decrypt the file

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task A>openssl enc -d -aes-128-cbc -in secret.enc -out secret_decrypted.txt -pass pass:mypassword
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Verify it matches the original

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task A>fc secret.txt secret_decrypted.txt
Comparing files secret.txt and SECRET_DECRYPTED.TXT
FC: no differences encountered
```

Task 2– ECC Signature Verification

Task 2A

Generate a private key

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>openssl ecparam -name prime256v1 -genkey -noout -out ecc_private.pem
```

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>type ecc_private.pem
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIHBywpA9MIruVxCp2Y4oDQvTEe3DnMb4IldFPY+PUERNoAoGCCqGSM49
AwEHoUQDQgAEyNiAxpGJ2+rkzMM7LS9Hjan22WPtG4AsrMtvt1QYjXVm9MpTsZ8F
artloEkrBVvYG5sKFIeAZ+4xcxOfi/6ypg==
-----END EC PRIVATE KEY-----
```

Extract the public key

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>openssl ec -in ecc_private.pem -pubout -out ecc_public_key.pem
```

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>type ecc_public_key.pem
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEyNiAxpGJ2+rkzMM7LS9Hjan22WPt
G4AsrMtvt1QYjXVm9MpTsZ8FartloEkrBVvYG5sKFIeAZ+4xcxOfi/6ypg==
-----END PUBLIC KEY-----
```

Task 2B

Create a text file

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>echo "Elliptic Curves are efficient." > ecc.txt
```

Sign the message using private key

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>openssl dgst -sha256 -sign ecc_private.pem -out ecc_signature.bin ecc.txt
```

Verify the signature using public key

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 2 - ECC>openssl dgst -sha256 -verify ecc_public_key.pem -signature ecc_signature.bin ecc.txt
Verified OK
```

Task 3– Hashing & HMAC

Task 3A

Creating data.txt

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>echo "Never trust, always verify." > data.txt
```

Hash using OpenSSL CLI

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>openssl dgst -sha256 data.txt
SHA2-256(data.txt)= b46512f3a09bcb3515aefd67fe6f1e4b3541829d6dce933221c637a1504b55ae
```

Task 3B

Creating HMAC

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>openssl dgst -sha256 -hmac "secretkey123" data.txt
HMAC-SHA2-256(data.txt)= 5080b9acd57d3c9a104d7f8502d689c02dabb1843c47a3752a7f7c72a5896072
```

Task 3C

Changing one letter in data.txt (I change “.”with symbol “!”)

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>echo "Never trust, always verify!" > data_modified.txt
```

Recomputing HMAC for modified file

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>openssl dgst -sha256 -hmac "secretkey123" data_modified.txt
HMAC-SHA2-256(data_modified.txt)= 7665f89eea13a0b428c3781326f7fcaf6bd57266df4de6a70e33697adf232c18
```

conclusion

The original HMAC:

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>openssl dgst -sha256 -hmac "secretkey123" data.txt
HMAC-SHA2-256(data.txt)= 5080b9acd57d3c9a104d7f8502d689c02dabb1843c47a3752a7f7c72a5896072
```

The modified HMAC:

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 3 - HH>openssl dgst -sha256 -hmac "secretkey123" data_modified.txt
HMAC-SHA2-256(data_modified.txt)= 7665f89eea13a0b428c3781326f7fcaf6bd57266df4de6a70e33697adf232c18
```

Even though we changed only one character (period to exclamation mark), the entire HMAC output changed completely. This is due to the avalanche effect in cryptographic hash functions.

Why HMAC is important:

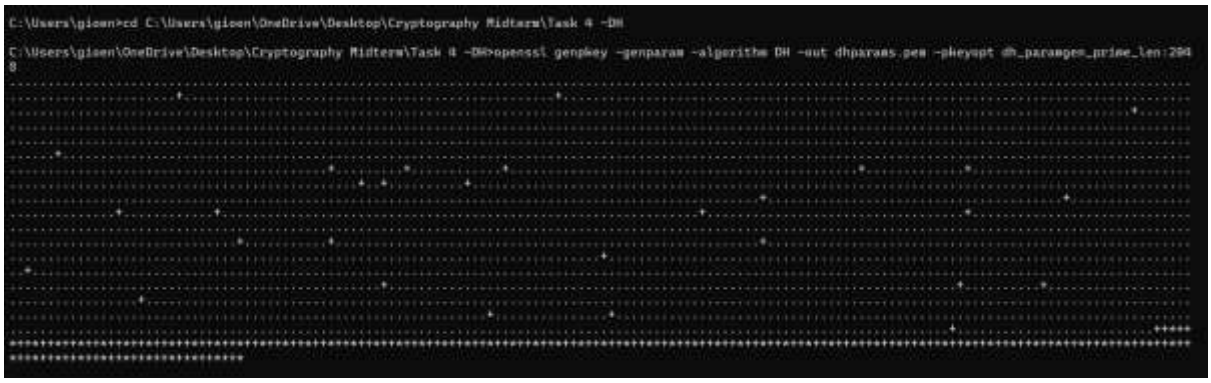
1. Integrity Verification: HMAC ensures that the message hasn't been tampered with during transmission or storage
2. Authentication: Only parties with the secret key can generate valid HMACs
3. Security: Even tiny changes in input produce completely different outputs
4. Tamper Detection: Any modification of the data or key results in a different HMAC value
5. Applications: Used in API security, message authentication, digital signatures, and secure communications

This demonstrates that HMAC provides both authentication and integrity protection, making it essential for secure data transmission and verification.

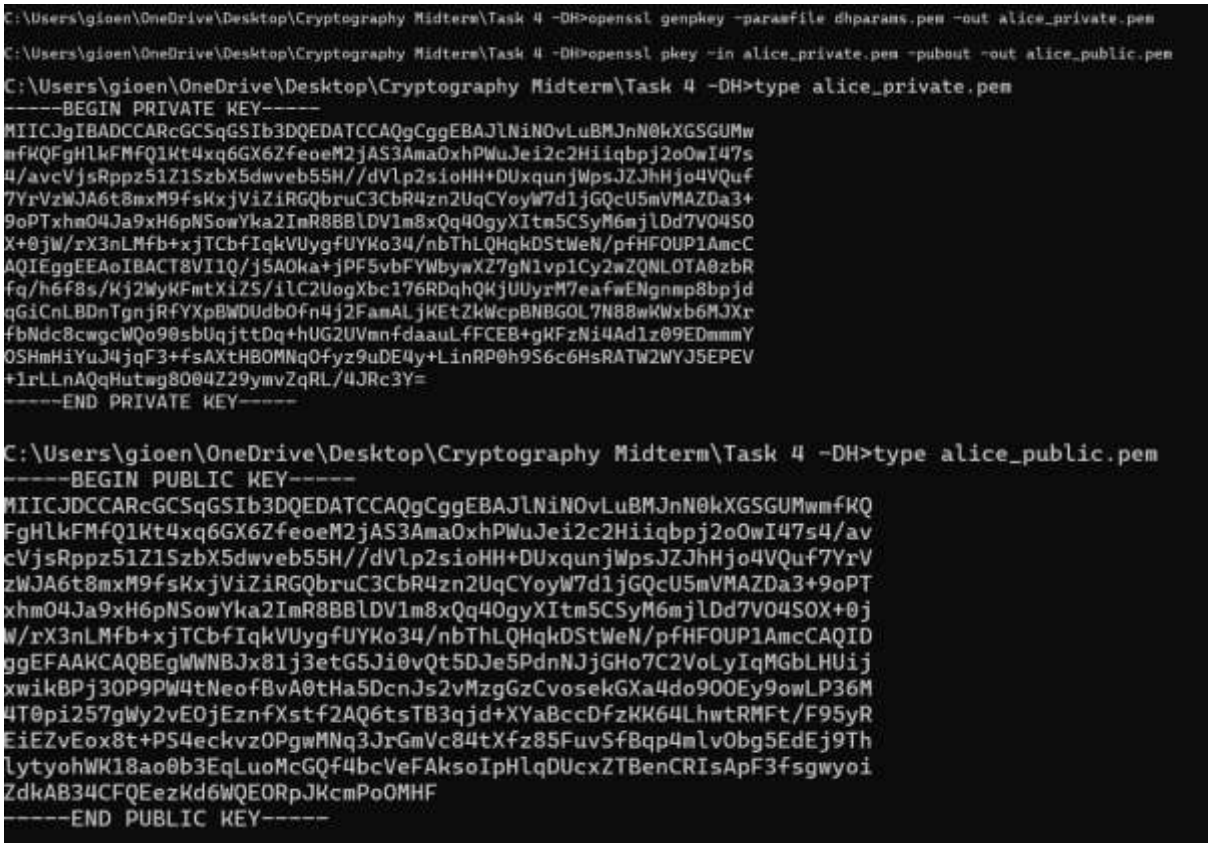
Task 4– Diffie-Hellman Key Exchange

Task 4A

Lets Generate DH parameters



Generating Alice's private/public key pair



Bob’s private/public key pair

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl genpkey -paramfile dhparams.pem -out bob_private.pem

C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl pkey -in bob_private.pem -pubout -out bob_public.pem
```

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>type bob_private.pem
-----BEGIN PRIVATE KEY-----
MIICJgIBADCCARcGCSqGSIb3DQEDATCCAQgCggEBAJlNiNOvLuBMJnN0kXGSGUMw
mfKQFgHlkFMfQ1Kt4xq6GX6ZfeoeM2jAS3AmaOxhPWuJei2c2Hiiqbpj2oOwI47s
4/avcVjsRppz51Z1SzbX5dwveb55H//dVlp2sioHH+DUxqunjWpsJZJhHjo4VQuf
7YrVzWJA6t8mxM9fsKxjViZiRGQbruC3CbR4zn2UqCYoyW7d1jGQcU5mVMAZDa3+
9oPTxhmO4Ja9xH6pNSowYka2ImR8BBlDV1m8xQq4OgyXItm5CSyM6mjlDd7VO4SO
X+0jW/rX3nLMfb+xjTCbfIqkVUygfUYKo34/nbThLQHqkDStWeN/pfHFOUP1AmcC
AQIEggEEAoIBACRQJjGI0HtUqKrhBDeW0B+POVvaLKCHim45k11VOsAjtN8Gq8mQ
OmvFK1N+se8iw4hAE6lJKGZ9Dl72B7vePa5ukrhCyqRjtVAs73PUcUPBR/V3yW8w
cCQ4Ez2sbalISlZVPDlaQvAd1096i+RvQeHeuH3h+iIuys9uk0mcemNsgz5eboqM
f1l26O3PlmyRDBz+ip/iiMOu4iykIGhs45c1CF2lk9Ccmjyhi3N4cuJMM0j1eGou
MBodzFG8Y6DpnALKn/mfo5LETRA8dbd+voIuOpWBuHof/W1UgZnvNTMYYBe7hDEG
bzOEGxw/53ocYLiJf4u4pvKd+Wk625PGeR4=
-----END PRIVATE KEY-----

C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>type bob_public.pem
-----BEGIN PUBLIC KEY-----
MIICJTCCARcGCSqGSIb3DQEDATCCAQgCggEBAJlNiNOvLuBMJnN0kXGSGUMwmfKQ
FgHlkFMfQ1Kt4xq6GX6ZfeoeM2jAS3AmaOxhPWuJei2c2Hiiqbpj2oOwI47s4/av
cVjsRppz51Z1SzbX5dwveb55H//dVlp2sioHH+DUxqunjWpsJZJhHjo4VQuf7YrV
zWJA6t8mxM9fsKxjViZiRGQbruC3CbR4zn2UqCYoyW7d1jGQcU5mVMAZDa3+9oPT
xhmO4Ja9xH6pNSowYka2ImR8BBlDV1m8xQq4OgyXItm5CSyM6mjlDd7VO4SOX+0j
W/rX3nLMfb+xjTCbfIqkVUygfUYKo34/nbThLQHqkDStWeN/pfHFOUP1AmcCAQID
ggEGAAKCAQEAhRgVEcf2lI8jLSUJYg05jPfFIazYMY+/z7Ebl5g3GF1jvS3T4qn+
rp+2yXj+QGm86Q9KMkD7B0lHs66L2N+41gp6SkP4KngL8DQFK3M2L7vG6K20Yz5q
84gd2f24yfJe9ZMMvKNzuBaxXjyKRR6rpHtgFzXRiDphZ53ZCxnN3UA8+AjVK0Yx
BXNj/uoAhG5HUqWyWiCJSEutosutH1Z9Pyu/X9nMBa+eZkOKOgv4eYAG5tXGMrg2
DXrMUdljFNWM1sRL5+fnVsGtPMqMISv8054RA7O03sRoUWpVprFET1Ogp7bDUaa/
zyt8UHVfK32/vwtwrO2JlATA5k2l5ufKAg==
-----END PUBLIC KEY-----
```

Derive shared secret for Alice

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl pkeyutl -derive -inkey alice_private.pem -peerkey bob_public.pem -out alice_shared_secret.bin
```

Derive shared secret for Bob

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl pkeyutl -derive -inkey bob_private.pem -peerkey alice_public.pem -out bob_shared_secret.bin
```

Verifing shared secrets matches

```
C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl dgst -sha256 alice_shared_secret.bin
SHA2-256(alice_shared_secret.bin)= 93c01abdac2bfa7348d78fc1e4dedca89459b63b8114fef35f07dccb9f4917a4

C:\Users\gioen\OneDrive\Desktop\Cryptography Midterm\Task 4 -DH>openssl dgst -sha256 bob_shared_secret.bin
SHA2-256(bob_shared_secret.bin)= 93c01abdac2bfa7348d78fc1e4dedca89459b63b8114fef35f07dccb9f4917a4
```

Task 4B

The Diffie-Hellman key exchange is fundamental to modern secure communication and is widely used in practice. In the TLS/SSL handshake that secures HTTPS connections, Diffie-Hellman (specifically in Ephemeral mode as DHE or ECDHE) enables web browsers and servers to establish a shared secret over an insecure channel without pre-shared keys. This provides forward secrecy, meaning even if a server's private key is compromised later, past communications remain secure.

The protocol is also crucial in secure messaging applications like Signal, WhatsApp, and Telegram, where it forms the basis of their end-to-end encryption. In the Signal Protocol, an extended triple Diffie-Hellman (X3DH) handshake allows users to establish secure sessions while providing identity verification and forward secrecy. Additionally, VPN protocols such as IPsec (using IKE) and SSH connections rely on Diffie-Hellman to securely negotiate session keys for encrypted tunnels.

Importance for Secure Communication**:**

Diffie-Hellman is essential because it solves the key distribution problem - it allows two parties who have never met to establish a shared secret over an insecure channel. This enables perfect forward secrecy, protects against eavesdropping, and forms the foundation for secure symmetric encryption in virtually all modern encrypted communications, ensuring confidentiality and integrity in our digital interactions.