HW1-test

2. OpenSSL Encryption.

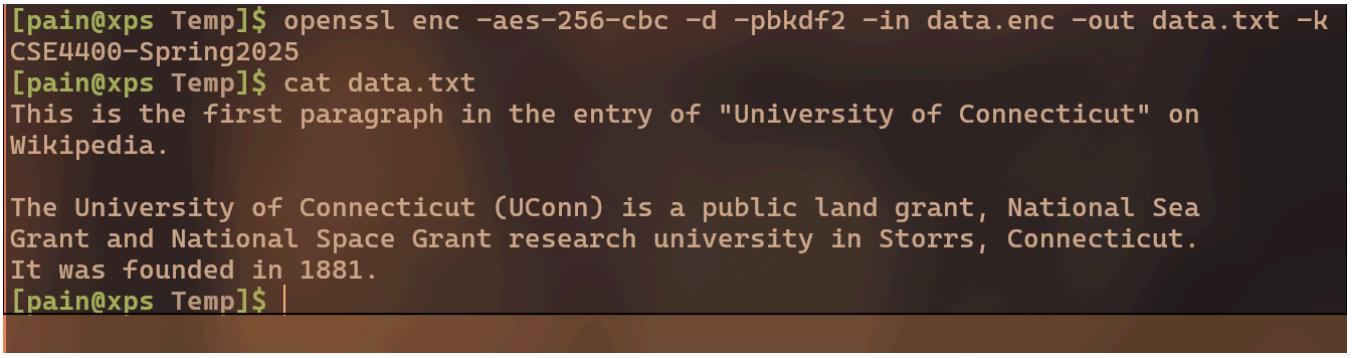
a) “data.enc” is a file that contains the ciphertext encrypted with AES-256-CBC. The passphrase is “CSE4400-Spring2025” (without quotation marks). The key derivation option is “-pbkdf2”. Use OpenSSL to decrypt the file and save the plaintext into “data.txt”. What is the command line you use to decrypt the file? What is the number near the end of the plaintext?

Answer:

I have used the below command to decrypt the file `data.enc`

```
openssl enc -aes-256-cbc -d -pbkdf2 -in data.enc -out data.txt -k CSE4400-Spring2025
```

The number near the end of the plain text is `1881`.



**b) “data.enc” is a binary file that some applications do not handle very well. Alternatively, OpenSSL can save the ciphertext in base64 format, which can be viewed with text editors. Encrypt the plaintext file with the parameters listed below and save the ciphertext in base64 format in file “data.asc”.

```
Algorithm: AES-128-CBC
Key (-K): ebf01882090b2e6cf9b7e61b0ae5e3fb (in hex)
Initialization vector: 5b2f79fbd76b65de872f5abfb8b5375e (in hex)
Output encoding: -base64
```

Note that openssl options are case sensitive. For example, -p and -P are different. Some options are given below. Find out other options in the manual.**

Answer:

Encryption:
I have used the `data.txt` file to encode with the given parameters using the below command.

```
openssl enc -aes-256-cbc -K ebf01882090b2e6cf9b7e61b0ae5e3fb  -iv 5b2f79fbd76b65de872f5abfb8b5375e  -base64 -
in data.txt -out data.asc
```

```
[pain@xps Temp]$ openssl enc -aes-256-cbc -K ebf01882090b2e6cf9b7e61b0ae5e3fb  -iv 5b2f79fbd76b65de872f5a
bfb8b5375e  -base64 -in data.txt -out data.asc
hex string is too short, padding with zero bytes to length
[pain@xps Temp]$ cat data.asc
76K753FT2K+vXLFhbLwXOpvoIf7a8LtgynReQmCbBTpmhJD9zPFJN3FMBOpuOWiP
0UBxQakBxRB4ZqXSQmVaAtczmrZGswIht7KKb4ot1cN3b5J/c2b2tV0EcD4Mai5W
JU/vVHHt0ytUnG1tt4e0Nm3kNB5zQl1X1JGwC15X45N6AyEHqmi8HXsw+YEVOhsN
CH7v7KKNJU3bTWzW9fC7jg8tHzfJZsbcyIO1JPHdiXEgiMDotZdm8svC4LbtG1T/
5jLtpWgX3TGMHmyA+j97id2vfVZzuH/QFbReu/ObE6XXKALT/+p5SmsQ9OoueIKY
hJzaUqEqzy6JBRxlGYGs/kQg1rPVxo6b/0nlT+MN/5Y=
[pain@xps Temp]$
```

---

3. OpenSSL Hash.

a) What is the SHA256 hash value of the file “data.enc”? The last 4 hex digits are c62d.

Answer:
The SHA256 has value is:
37a2471f1b23e85d3eb73226d7ceaf6f1ae80af1f974bf17f292d8fb13d8c62d

```
[pain@xps Temp]$ openssl dgst -sha256 data.enc
SHA2-256(data.enc)= 37a2471f1b23e85d3eb73226d7ceaf6f1ae80af1f974bf17f292d8fb13d8c62d
```

```
[pain@xps Temp]$ openssl dgst -sha256 data.enc
SHA2-256(data.enc)= 37a2471f1b23e85d3eb73226d7ceaf6f1ae80af1f974bf17f292d8fb13d8c62d
[pain@xps Temp]$
```

b) What is the SHA256 hash value of “data.txt”? The last 4 hex digits are 172c.

Answer:
3f5976523af6e908bbe981112277d57a63d84f3b2752d268ac969270df91172c

```
openssl dgst -sha256 data.txt
SHA2-256(data.txt)= 3f5976523af6e908bbe981112277d57a63d84f3b2752d268ac969270df91172c
```

```
[pain@xps Temp]$ openssl dgst -sha256 data.txt
SHA2-256(data.txt)= 3f5976523af6e908bbe981112277d57a63d84f3b2752d268ac969270df91172c
[pain@xps Temp]$
```

c) What is the MD5 hash of “data.asc”? The last 4 hex digits are 94fd.

Answer:
3f47ed3b706c500d72a7b0aa9e7bb09d

```
openssl dgst -md5 data.asc
MD5(data.asc)= 3f47ed3b706c500d72a7b0aa9e7bb09d
```

```
[pain@xps Temp]$ openssl dgst -md5 data.asc
MD5(data.asc)= 3f47ed3b706c500d72a7b0aa9e7bb09d
[pain@xps Temp]$ openssl --version
OpenSSL 3.4.0 22 Oct 2024 (Library: OpenSSL 3.4.0 22 Oct 2024)
[pain@xps Temp]$ uname -a
Linux xps 6.12.11-1-MANJARO #1 SMP PREEMPT_DYNAMIC Thu, 23 Jan 2025 20:29:02 +0000 x86_64 GNU/Linux
[pain@xps Temp]$
```

4. OpenSSL Public-key. We experiment with public key and digital signature in this problem.

a) "private.pem" is an RSA private key file. We can extract the public key from it. Use OpenSSL to save the public key in "public.pem".

Answer:

```
openssl rsa -in private.pem -pubout -out public.pem
```

```
[pain@xps Temp]$ openssl rsa -in private.pem -pubout -out public.pem
writing RSA key
[pain@xps Temp]$ cat public.pem
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0sG0q3QyUAlF7QYrRURT
69XAsjjsKrpEzVT0x+hH8SyqhEgJo2U4VUk2qrGrSrADDvO1DfQ8TRkCn/+5NDzI
g/AqvNaNsYukoc4a5Fm81gla2SrUM+VrapxyTEOLuqIGICRMwpa1lHjnbhSJES1Q
w1PWv4qFVNKrarlxm6+nxgwNIsVrTRihPQ+4wVbkGUJFqOwr/VOkq0mijJv8CYLg
KWtdWtElK9B9xNqhKg8R4Q0BM9Wu03lq00/rf083MS6nwpMcJ0JrxhvUbRdcrrRH
n1gu5YwrzWxSBK75UjXwB6aFaSzo3WS8GC0eYZSAC90iCGqzpT+qr6chyfYu6KYE
BQIDAQAB
-----END PUBLIC KEY-----
[pain@xps Temp]$
```

b) "small.rsa.enc" is a file encrypted with "public.pem". Decrypt it with the private key and save the plaintext in "small.txt".

Answer:

```
openssl pkeyutl -decrypt -inkey private.pem -in small.rsa.enc -out small.txt
```

```
[pain@xps Temp]$ openssl pkeyutl -decrypt -inkey private.pem -in small.rsa.enc -out small.txt
[pain@xps Temp]$ cat small.txt
◆◆1ce336ed32d8f523633686f1b3ae8576952fe7622b4fda25118c8651c316b359
[pain@xps Temp]$
```

c) Use the digt command to sign "data.enc" and then verify the signature. The hash algorithm is SHA256. The sign command looks like

`openssl dgst -sign private.pem …`

Answer:

To Sign `data.enc`, I used the following command:

```
openssl dgst -sha256 -sign private.pem -out data.enc.sig data.enc
```

To verify the Signature:

```
openssl dgst -sha256 -verify public.pem -signature data.enc.sig data.enc
```