1 Information

ID	Name
21120566	Huu Thuan Nguyen

2 Setting up project

```
cd Source
conda create -n openssl_introduction
conda activate openssl_introduction
conda install --file requirements.txt
python key_generation.py      # for exercise 1
python encrypt_and_decrypt.py # for exercise 2
python sign_and_verify.py     # for exercise 3
```

> Có thể thay thế Miniconda bằng các Python Virtual Environment khác, hoặc tải và chạy trực tiếp bằng `python`.

3 Exercises

3.1 (4 pts.) RSA key generation

3.1.1 Private key

Để tạo một private key có độ dài 512 bit và lưu vào file `priv.pem`, ta dùng lệnh:

```
$ openssl genpkey \
    -out priv.pem \
    -algorithm RSA \
    -pkeyopt rsa_keygen_bits:512
```

Kết quả của file `priv.pem`:

```
-----BEGIN PRIVATE KEY-----
MIIBVQIBADANBgkqhkiG9w0BAQEFAASCAT8wggE7AgEAAkEAsR0LmwBLLkOcx1r5
oKd29QntwozQssSmL16q8XpN2H56maEn+vbxX9hmkLoz3PSkahprG6BsRTTceLhh
P+2xHwIDAQABAkBDLx1j4qyiFAu4o43lnyDEustx7TCNX4ManPKDPz2gTVhm1thA
jDQJyKY+StBpgJhoqzUtqRI326azGCGyO1/BAiEA2ZHHoGl1aa1yYW583/g7G4uW
pRDTsm9Z4Kltx7TWcVkCIQDQZeVRx5WqkJ12YLcErjgYCPHwisGF930qiGoPsewv
NwIgHUmIa8mGaqFUglUTLjsGCeEyeCnZtW1T1fA4vOdU7pECIQCHHKky/k1gymXs
rAO8ux+PJa2GoM+bhbvW6Z6qElbovQIhALmhLAQ/pyrR27nWfzriqfwNUmZST+ef
jXPWlRdjUZ30
-----END PRIVATE KEY-----
```

File này có định dạng **PEM**, một tiêu chuẩn để lưu trữ các key và certificate, được mã hoá bằng **base64**.

Dòng đầu tiên và cuối cùng của file là tên của certificate, ở đây là `PRIVATE KEY`.

Key trong file này được lưu trữ dưới dạng **PKCS #8**, một tiêu chuẩn để lưu trữ các private key.

Ưu điểm của **PKCS #8** là ngoài việc lưu trữ private key, nó còn hỗ trợ mã hoá bằng **passphrase**, một chuỗi các từ tiếng Anh có vai trò như mật khẩu và được dùng phổ biến trong việc lưu trữ trên các ví blockchain hiện nay.

Để giải mã file **PKCS #8** này, ta dùng lệnh:

```
$ openssl rsa \
    -in priv.pem \
    -text \
    -noout
```

Console sẽ hiển thị thông tin của private key như sau:

```
> Private-Key: (512 bit, 2 primes)
modulus:
    00:b1:1d:0b:9b:00:4b:2e:43:9c:c7:5a:f9:a0:a7:
    76:f5:09:ed:c2:8c:d0:b2:c4:a6:2f:5e:aa:f1:7a:
    4d:d8:7e:7a:99:a1:27:fa:f6:f1:5f:d8:66:90:ba:
    33:dc:f4:a4:6a:1a:6b:1b:a0:6c:45:34:dc:78:b8:
    61:3f:ed:b1:1f
publicExponent: 65537 (0x10001)
privateExponent:
    43:2f:1d:63:e2:ac:a2:14:0b:b8:a3:8d:e5:9f:20:
    c4:ba:cb:71:ed:30:8d:5f:83:1a:9c:f2:83:3f:3d:
    a0:4d:58:66:d6:d8:40:8c:34:09:c8:a6:3e:4a:d0:
    69:80:98:68:ab:35:2d:a9:12:37:db:a6:b3:18:21:
    b2:3b:5f:c1
prime1:
    00:d9:91:c7:a0:69:75:69:ad:72:61:6e:7c:df:f8:
    3b:1b:8b:96:a5:10:d3:b2:6f:59:e0:a9:6d:c7:b4:
    d6:71:59
prime2:
    00:d0:65:e5:51:c7:95:aa:90:9d:76:60:b7:04:ae:
    38:18:08:f1:f0:8a:c1:85:f7:7d:2a:88:6a:0f:b1:
    ec:2f:37
exponent1:
    1d:49:88:6b:c9:86:6a:a1:54:82:55:13:2e:3b:06:
    09:e1:32:78:29:d9:b5:6d:53:d5:f0:38:bc:e7:54:
    ee:91
exponent2:
    00:87:1c:a9:32:fe:4d:60:ca:65:ec:ac:03:bc:bb:
    1f:8f:25:ad:86:a0:cf:9b:85:bb:d6:e9:9e:aa:12:
    56:e8:bd
coefficient:
    00:b9:a1:2c:04:3f:a7:2a:d1:db:b9:d6:7f:3a:e2:
    a9:fc:0d:52:66:52:4f:e7:9f:8d:73:d6:95:17:63:
    51:9d:f4
```

Như vậy, private key trên, ta sẽ đọc được các thông tin theo thứ tự là:

- Modulus: n
- Public exponent: e
- Private exponent: d
- Prime 1: p
- Prime 2: q
- Exponent 1: $d \mod (p-1)$
- Exponent 2: $d \mod (q-1)$
- Coefficient: $q^{-1} \mod p$

3.1.2 Public key

Để tạo một public key và lưu vào file `pub.pem` từ private key từ `priv.pem`, ta dùng lệnh:

```
$ openssl pkey \
    -in priv.pem \
    -out pub.pem \
    -pubout
```

Kết quả của file `pub.pem`:

```
-----BEGIN PUBLIC KEY-----
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALEdC5sASy5DnMda+aCndvUJ7cKM0LLE
pi9eqvF6Tdh+epmhJ/r28V/YZpC6M9z0pGoaaxugbEU03Hi4YT/tsR8CAwEAAQ==
-----END PUBLIC KEY-----
```

Tương tự với private key, đây là một file **PEM** được mã hoá bằng **base64**.

Tuy nhiên, key trong file này được lưu trữ dưới dạng **X.509**, một tiêu chuẩn để lưu trữ các public key.

Để giải mã file **X.509** này, ta dùng lệnh:

```
$ openssl rsa \
    -in pub.pem \
    -pubin \
    -text \
    -noout
```

Console sẽ hiển thị thông tin của public key như sau:

```
> Public-Key: (512 bit)
Modulus:
    00:b1:1d:0b:9b:00:4b:2e:43:9c:c7:5a:f9:a0:a7:
    76:f5:09:ed:c2:8c:d0:b2:c4:a6:2f:5e:aa:f1:7a:
    4d:d8:7e:7a:99:a1:27:fa:f6:f1:5f:d8:66:90:ba:
    33:dc:f4:a4:6a:1a:6b:1b:a0:6c:45:34:dc:78:b8:
    61:3f:ed:b1:1f
Exponent: 65537 (0x10001)
```

Như vậy, public key trên, ta sẽ đọc được các thông tin theo thứ tự là:

- Modulus: n
- Public exponent: e

3.1.3 Code

Chạy file `key_generation.py`, console sẽ xuất ra các thông tin của private key và public key được tạo ra từ `priv.pem` và `pub.pem` trong thư mục `openssl_outputs`.

```
$ python key_generation.py
```

```
> ---MESSAGE---
Hello World!

---PRIVATE KEY---
n: 12144202980105029326780271137687098746341120315748737279660417039899117597625384106994513687327035
e: 65537
d: 73448551737556349455359507924455427164612811992513038368234730031963825658340877409427182254879973
p: 11418754032567399709460663534078166883952825565493102643776113942812064595399 3
```

```
q: 106353135775309447031522329646720602723835437848593458187713291455497952556963
dmp1: 731450498254744659051937250444993704251374902825832647341753573451932929153 69
dmq1: 534386493290956267566112320561897774950928630903792144608602573756740097609 1
iqmp: 103473184854957043546353952212899144956884896155893446361759393193746125755469

---PUBLIC KEY---
n: 121442029801050293267802711376870987463411203157487372796604170398991175976253841069945136873270 35
e: 65537
```

3.2 (3 pts.) RSA encryption and decryption

3.2.1 Encrypt

Có được public key từ `pub.pem` của người nhận, ta có thể mã hoá một message từ `message.txt` để gửi cho người đó bằng lệnh:

```
$ openssl pkeyutl \
    -in openssl_outputs/message.txt \
    -out openssl_outputs/encrypted_message.txt \
    -inkey openssl_outputs/pub.pem \
    -pubin \
    -encrypt
```

Với message là một plaintext như sau:

```
Hello World!
```

Message được mã hoá sẽ nằm trong file `encrypted_message.txt`, đây còn được gọi là ciphertext với định dạng **binary**:

```
21:ef:bf:bd:ef:bf:bd:3e:ef:bf:bd:ef:bf:bd:ef:bf:
bd:0a:6c:7a:6c:48:ef:bf:bd:1f:ef:bf:bd:ef:bf:bd:
43:71:1d:1b:26:ef:bf:bd:3f:34:ef:bf:bd:ef:bf:bd:
3d:35:37:ef:bf:bd:ef:bf:bd:ef:bf:bd:57:ef:bf:bd:
4d:ef:bf:bd:4b:ef:bf:bd:ef:bf:bd:ef:bf:bd:ef:bf:
bd:5b:ef:bf:bd:ef:bf:bd:1e:ef:bf:bd:ef:bf:bd:ef:
bf:bd:ef:bf:bd:ef:bf:bd:71:47:d9:b1:ef:bf:bd:57:
07:ef:bf:bd:45:3d:ef:bf:bd:ef:bf:bd:6e:6e
```

3.2.2 Decrypt

Với private key từ `priv.pem`, ta có thể giải mã message từ `encrypted_message.txt` của người gửi bằng lệnh:

```
$ openssl pkeyutl \
    -in openssl_outputs/encrypted_message.txt \
    -out openssl_outputs/decrypted_message.txt \
    -inkey openssl_outputs/priv.pem \
    -decrypt
```

Message được giải mã sẽ nằm trong file `decrypted_message.txt`, đây cũng là plaintext ban đầu:

```
Hello World!
```

3.2.3 Code

Chạy file `encrypt_and_decrypt.py`, console sẽ xuất ra các thông tin của encryped message và decrypted message được encrypt và decrypt bởi thư viện `cryptography`. Sau đó xuất ra file `encrypted_message.txt` trong thư mục `code_outputs`.

```
$ python encrypt_and_decrypt.py
```

```
> ---ENCRYPTED MESSAGE---
85787353848996262506768262251532399647679834063898441072493733611057098360083105872933735018143989980

---DECRYPTED MESSAGE---
Hello World!

---EXPORT ENCRYPTED MESSAGE---
Encrypted Message Exported Successfully
```

Ta sẽ thử dùng `openssl` để decrypt message đã được tạo ra từ file `encrypted_message.txt` và xuất ra file `decrypted_message.txt` trong thư mục `code_outputs`.

```
$ openssl pkeyutl \
    -in code_outputs/encrypted_message.txt \
    -out code_outputs/decrypted_message.txt \
    -inkey openssl_outputs/priv.pem \
    -decrypt
```

Plain text `openssl` giải mã được sẽ khớp hoàn toàn với message ban đầu:

```
Hello World!
```

3.3 (3 pts.) RSA signature and verification

3.3.1 Sign

Có được private key từ `priv.pem`, ta có thể tạo một chữ ký cho message từ `message.txt` bằng lệnh:

```
$ openssl pkeyutl \
    -in openssl_outputs/message.txt \
    -out openssl_outputs/signature.txt \
    -inkey openssl_outputs/priv.pem \
    -sign
```

Chữ ký được lưu trong file `signature.txt`, có định dạng **binary**:

```
12:1b:17:78:5e:12:e5:90:12:ca:e8:94:c9:75:fe:cf:
e9:27:8c:5f:26:95:7b:9d:88:40:58:ce:5c:91:f7:3d:
15:0d:3b:a4:b6:31:6d:42:7f:14:c2:c4:d4:bb:f9:81:
f1:27:06:12:00:f3:98:c9:dc:b8:84:a7:7e:8d:f9:48
```

3.3.2 Verify

Với public key từ `pub.pem`, ta có thể xác minh chữ ký của message từ `message.txt` bằng lệnh:

```
$ openssl pkeyutl \
    -in openssl_outputs/message.txt \
    -sigfile openssl_outputs/signature.txt \
    -inkey openssl_outputs/pub.pem \
```

```
    -pubin \
    -verify
```

Console sẽ hiển thị thông báo xác minh chữ ký thành công:

```
> Signature Verified Successfully
```

3.3.3 Code

Chạy file `sign_and_verify.py`, console sẽ xuất ra các thông tin của signature và kết quả verify được sign và verify bởi thư viện `cryptography`. Sau đó xuất ra file `signature.txt` trong thư mục `code_outputs`.

```
$ python sign_and_verify.py
```

```
> ---SIGNATURE---
526210259335184488108980042981407566432655229770816191390276893433116050224389694456438951781024166 10

---VERIFY---
Signature Verified Successfully

---EXPORT SIGNATURE---
Signature Exported Successfully
```

Ta sẽ thử dùng `openssl` để verify signature đã được tạo ra từ file `signature.txt` trong thư mục `code_outputs`.

```
$ openssl pkeyutl \
    -in openssl_outputs/message.txt \
    -sigfile code_outputs/signature.txt \
    -inkey openssl_outputs/pub.pem \
    -pubin \
    -rawin \
    -verify
```

Console sẽ hiển thị thông báo verify signature thành công:

```
> Signature Verified Successfully
```