Crypto_RSA LAB

40647027s 陳冠穎

Task1:

程式碼:

1. 先設定好給定的 p, q, e
2. 算出 n = p*q
3. 算出 Euler's totient function of p*q，這邊已知 p 與 q 為 prime，因此直接帶(p-1)(q-1)
4. 再算出 modular inverse of e mod r 即為 d

```
int main(){

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p = BN_new();
    BIGNUM *q = BN_new();
    BIGNUM *e = BN_new();
    BIGNUM *n = BN_new();
    BIGNUM *eular_pq = BN_new();
    BIGNUM *d = BN_new();

    BN_hex2bn(&p, "F7E75FDC469067FFDC4E847C51F452DF");
    BN_hex2bn(&q, "E85CED54AF57E53E092113E62F436F4F");
    //BN_dec2bn(&p, "11");
    //BN_dec2bn(&q, "13");
    BN_hex2bn(&e, "0D88C3");
    //BN_dec2bn(&e, "7");

    BN_mul(n, p, q, ctx);
    printBN("The p*q = n is ", n);

    EularPHI(eular_pq, p, q);
    printBN("The eularPHI(p*q) = r is ", eular_pq);

    BN_mod_inverse(d, e, eular_pq, ctx);
    printBN("The inverse e mod r = 1 is d is ", d);

    return 0;
}
```

其中 EularPHI function 的 implementation:

```
void EularPHI(BIGNUM *r, BIGNUM *p, BIGNUM *q){

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *p_minus_one = BN_new();
    BIGNUM *q_minus_one = BN_new();
    BIGNUM *one = BN_new();
    BN_dec2bn(&one, "1");
    BN_sub(p_minus_one, p, one);
    BN_sub(q_minus_one, q, one);
    BN_mul(r, p_minus_one, q_minus_one, ctx);

}
```

輸出結果:

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The p*q = n is  E103ABD94892E3E74AFD724BF28E78366D9676BCCC70118BD0AA1968DBB143D1
The eularPI(p*q) = r is  E103ABD94892E3E74AFD724BF28E78348D52298BD687C44DEB3A81065A7981A4
The inverse e mod r = 1 is d is  3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB
```

d:3587A24598E5F2A21DB007D89D18CC50ABA5075BA19A33890FE7C28A9B496AEB

Task2:

```
int main(){

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *pk_n = BN_new();
    BIGNUM *pk_e = BN_new();
    BIGNUM *sk_d = BN_new();
    BIGNUM *msg_M = BN_new();
    BIGNUM *cipher = BN_new();
    BIGNUM *plain = BN_new();


    BN_hex2bn(&pk_n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&pk_e, "010001");
    BN_hex2bn(&msg_M, "4120746f702073656372657421");
    BN_hex2bn(&sk_d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    // Task2
    // Encrypt msg_M, cipher = M^e mod N
    BN_mod_exp(cipher, msg_M, pk_e, pk_n, ctx);
    printBN("The cipher is ", cipher);

    // Decrypt cipher to plain, plain = cipher^d mod N
    BN_mod_exp(plain, cipher, sk_d, pk_n, ctx);
    printBN("The plain is ", plain);


    return 0;
}
```

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The cipher is  6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5CFC5FADC
The plain is  4120746F702073656372657421
gary@ubuntu:~/Desktop/hw2$ python -c 'print("4120746F702073656372657421".decode("hex"))'
A top secret!
gary@ubuntu:~/Desktop/hw2$
```

"A top secret!"加密過後：
6FB078DA550B2650832661E14F4F8D2CFAEF475A0DF3A75CACDC5DE5C
FC5FADC

Task3:

```
int main(){

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *pk_n = BN_new();
    BIGNUM *pk_e = BN_new();
    BIGNUM *sk_d = BN_new();
    BIGNUM *msg_M = BN_new();
    BIGNUM *cipher = BN_new();
    BIGNUM *plain = BN_new();

    BN_hex2bn(&pk_n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
    BN_hex2bn(&pk_e, "010001");
    BN_hex2bn(&cipher, "8C0F971DF2F3672B28811407E2DABBE1DA0FEBBBDFC7DCB67396567EA1E2493F");
    BN_hex2bn(&sk_d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");

    // Decrypt cipher to plain, plain = cipher^d mod N
    BN_mod_exp(plain, ciphser, sk_d, pk_n, ctx);
    printBN("The plain is ", plain);

    return 0;
}
```

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The plain is  50617373776F72642069732064656573
gary@ubuntu:~/Desktop/hw2$ python -c 'print("50617373776F72642069732064656573".decode("hex"))'
Password is dees
gary@ubuntu:~/Desktop/hw2$
```

解密過後明文為："Password is dees"。

Task4:

```
/*
gary@ubuntu:~$ python -c 'print("I owe you $2000.".encode("hex"))'
49206f776520796f752024323030302e
gary@ubuntu:~$ python -c 'print("I owe you $3000.".encode("hex"))'
49206f776520796f752024333030302e
*/
BN_hex2bn(&pk_n, "DCBFFE3E51F62E09CE7032E2677A78946A849DC4CDDE3A4D0CB81629242FB1A5");
BN_hex2bn(&pk_e, "010001");
BN_hex2bn(&sk_d, "74D806F9F3A62BAE331FFE3F0A68AFE35B3D2E4794148AACBC26AA381CD7D30D");
BN_hex2bn(&msg_M1, "49206f776520796f752024323030302e");
BN_hex2bn(&msg_M2, "49206f776520796f752024333030302e");

// Sign with sk_n
BN_mod_exp(cipher, msg_M1, sk_d, pk_n, ctx);
printBN("The cipher1 is ", cipher);

BN_mod_exp(cipher, msg_M2, sk_d, pk_n, ctx);
printBN("The cipher2 is ", cipher);
```

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The cipher1 is  55A4E7F17F04CCFE2766E1EB32ADDBA890BBE92A6FBE2D785ED6E73CCB35E4CB
The cipher2 is  BCC20FB7568E5D48E434C387C06A6025E90D29D848AF9C3EBAC0135D99305822
```

兩段訊息在 16 進位表示之下只差了一個 byte，但在經過 RSA 簽章之後，簽章完的訊息截然不同。

Task5:

```
BN_hex2bn(&pk_n, "AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115");
BN_hex2bn(&pk_e, "010001");
BN_hex2bn(&sign_S1, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F");
BN_hex2bn(&sign_S2, "643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6803F");

// Verify with pk_n, pk_e
BN_mod_exp(plain, sign_S1, pk_e, pk_n, ctx);
printBN("The plain1 is ", plain);

BN_mod_exp(plain, sign_S2, pk_e, pk_n, ctx);
printBN("The plain2 is ", plain);
```

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The plain1 is  4C61756E63682061206D697373696C652E
The plain2 is  91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294
gary@ubuntu:~/Desktop/hw2$ python -c 'print("4C61756E63682061206D697373696C652E".decode("hex"))'
Launch a missile.
gary@ubuntu:~/Desktop/hw2$ python -c 'print("91471927C80DF1E42C154FB4638CE8BC726D3D66C83A4EB6B7BE0203B41AC294".decode("hex"))'
��,O�c��rm=f�:N����
gary@ubuntu:~/Desktop/hw2$
```

此簽章確實為 Alice 簽的發的，因為使用 Alice 的公鑰驗證之後，與 Alice 發的訊息一致。

就算簽章訊息只有一個 bit 被修改例如題目要求尾端 2F 改為 3F，驗章出來的結果就會大不相同，長度甚至還不一樣了。

Task6 在下一頁…

Task6:

Step1：我嘗試的網站為 https://aws-demo.itm.monster:4430，為之前實習公司的伺服器網站，當初我有使用 Let's Encrypt 作為網域的 CA 認證，現在拿來自己實驗覺得很有趣。

```
gary@ubuntu:~$ openssl s_client -connect aws-demo.itm.monster:4430 -showcerts
CONNECTED(00000005)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
verify return:1
depth=0 CN = *.itm.monster
verify return:1
---
Certificate chain
 0 s:CN = *.itm.monster
   i:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
-----BEGIN CERTIFICATE-----
MIIFUzCCBDugAwIBAgISA2aXo5Nrrd7E3FJjTwH8zBTCMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0yMDAyMTAwMDI3MzJaFw0y
MDA1MTAwMDI3MzJaMBgxFjAUBgNVBAMMDSouaXRtLm1vbnN0ZXIwggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDFPJMQ346g8BDJNJxRKe+00TYwk714j4WG
7C3nEZ+vja8urtxMipxTk6+Brg+Qj5g93b6jqJcxDiLU7lyqc+k/auTlG1Rqt9Ke
htiW5XajWLLih3yrWg0FRgVP6M9Fulvx9akgXm9pLMrrbVbwA2kUJSV0UGgvYq4G
HgxmTRd4+g36XNM65JxIdPza9qfzw4YUFPrJI1ZHOH+DzBE8L0iFdcTmU/OMzV0I
p2z1gZsWWjPEK1dol+nxBI+wNlvPPMXJe8HL3nUSvIjeyHf2fr1hDV9YO94mo7WJ
Qeu8e4udwpEVqJNQ/y4IaXfJVOP+0UhUZBEl1tMM3smdcoJ3qF7jAgMBAAGjggJj
MIICXzAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUF
BwMCMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFDbgHPO3xEmWItoAsLu05n/Y4hm7
MB8GA1UdIwQYMBaAFKhKamMEfd265tE5t6ZFZe/zqOyhMG8GCCsGAQUFBwEBBGMw
YTAuBggrBgEFBQcwAYYiaHR0cDovL29jc3AuaW50LXgzLmxldHNlbmNyeXB0Lm9y
ZzAvBggrBgEFBQcwAoYjaHR0cDovL2NlcnQuaW50LXgzLmxldHNlbmNyeXB0Lm9y
Zy8wGAYDVR0RBBEwD4INKi5pdG0ubW9uc3RlcjBMBgNVHSAERTBDMAgGBmeBDAEC
ATA3BgsrBgEEAYLfEwEBATAoMCYGCCsGAQUFBwIBFhpodHRwOi8vY3BzLmxldHNl
bmNyeXB0Lm9yZzCCAQUGCisGAQQB1nkCBAIEgfYEgfMA8QB3AOcS8rA3fhpi+47J
DGGE8ep7N8tWHREmW/Pg80vyQVRuAAABcCy1a+oAAAQDAEgwRgIhAPdawn4Jcj0E
0H7JrP8dmLORFQ3AQmcWqmM2ybx6au+GAiEAxeXB+Nm2n9yJ1Xpc06uSQO1g9vYb
H7OSAqzgmTC/S/kAdgCyHgXMi6LNiiBOh2b5K7mKJSBna9r6cOeySVMt74uQXgAA
AXAstWviAAAEAwBHMEUCIQCP5wiuD1DNCQAWhUIkhRpM9R0VVNHiDWb1c/qBwMLD
AwIgToEqjXGy06ZaZS1oTfrIKtIJwJbgYpe4CLWkoUAIGTswDQYJKoZIhvcNAQEL
BQADggEBABR0f/bbQA12A54WmxbayPnXqeW9fHrxJV+vbP989ztPAjGwLd9WnjuJ
HsWFjp86pvNlBPRQJlmGKm+bl3YHsvTf/K66sdP4A22UbJZiFfV/pH2SkKGSQJgt
IpTrGbrFARGlrqlgoPLbTKqd+O7f59B1j9plGGzze2dSnCOZMHR3wcvZcUWlMG7y
UujHnkR9LEzMyxudWprwZnIPplP5lR4JNNMWfqyZfifrWi7wKQrQbSBXuLefNlEQ
IMlZaHSU4hOk0el/PGiyuFXKc1IjPYYqVDGEnHFHMvevxKtl0ucNarJR9vEar1bl
U71GKpD+LiN+tUFY9qhxYlPR7aPq+1g=
-----END CERTIFICATE-----
 1 s:C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
   i:O = Digital Signature Trust Co., CN = DST Root CA X3
-----BEGIN CERTIFICATE-----
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
```

Step2：找到 Public Key N and e

```
gary@ubuntu:~/Desktop/hw2$ openssl x509 -in c1.pem -noout -modulus
Modulus=9CD30CF05AE52E47B7725D3783B3686330EAD735261925E1BDBE35F170922FB7B84B4105ABA99E350858ECB12AC
468870BA3E375E4E6F3A76271BA7981601FD7919A9FF3D0786771C8690E9591CFFEE699E9603C48CC7ECA4D7712249D471B
5AEBB9EC1E37001C9CAC7BA705EACE4AEBBD41E53698B9CBFD6D3C9668DF232A42900C867467C87FA59AB8526114133F65E
98287CBDBFA0E56F68689F3853F9786AFB0DC1AEF6B0D95167DC42BA065B299043675806BAC4AF31B9049782FA2964F2A20
252904C674C0D031CD8F31389516BAA833B843F1B11FC3307FA27931133D2D36F8E3FCF2336AB93931C5AFC48D0D1D64163
3AAFA8429B6D40BC0D87DC393
```

```
    Public Key Algorithm: rsaEncryption
        RSA Public-Key: (2048 bit)
        Modulus:
            00:9c:d3:0c:f0:5a:e5:2e:47:b7:72:5d:37:83:b3:
            68:63:30:ea:d7:35:26:19:25:e1:bd:be:35:f1:70:
            92:2f:b7:b8:4b:41:05:ab:a9:9e:35:08:58:ec:b1:
            2a:c4:68:87:0b:a3:e3:75:e4:e6:f3:a7:62:71:ba:
            79:81:60:1f:d7:91:9a:9f:f3:d0:78:67:71:c8:69:
            0e:95:91:cf:fe:e6:99:e9:60:3c:48:cc:7e:ca:4d:
            77:12:24:9d:47:1b:5a:eb:b9:ec:1e:37:00:1c:9c:
            ac:7b:a7:05:ea:ce:4a:eb:bd:41:e5:36:98:b9:cb:
            fd:6d:3c:96:68:df:23:2a:42:90:0c:86:74:67:c8:
            7f:a5:9a:b8:52:61:14:13:3f:65:e9:82:87:cb:db:
            fa:0e:56:f6:86:89:f3:85:3f:97:86:af:b0:dc:1a:
            ef:6b:0d:95:16:7d:c4:2b:a0:65:b2:99:04:36:75:
            80:6b:ac:4a:f3:1b:90:49:78:2f:a2:96:4f:2a:20:
            25:29:04:c6:74:c0:d0:31:cd:8f:31:38:95:16:ba:
            a8:33:b8:43:f1:b1:1f:c3:30:7f:a2:79:31:13:3d:
            2d:36:f8:e3:fc:f2:33:6a:b9:39:31:c5:af:c4:8d:
            0d:1d:64:16:33:aa:fa:84:29:b6:d4:0b:c0:d8:7d:
            c3:93
        Exponent: 65537 (0x10001)
```

Step3: 輸出 Server 憑證中的簽名

```
Signature Algorithm: sha256WithRSAEncryption
     14:74:7f:f6:db:40:0d:76:03:9e:16:9b:16:da:c8:f9:d7:a9:
     e5:bd:7c:7a:f1:25:5f:af:6c:ff:7c:f7:3b:4f:02:31:b0:2d:
     df:56:9e:3b:89:1e:c5:85:8e:9f:3a:a6:f3:65:04:f4:50:26:
     59:86:2a:6f:9b:97:76:07:b2:f4:df:fc:ae:ba:b1:d3:f8:03:
     6d:94:6c:96:62:15:f5:7f:a4:7d:92:90:a1:92:40:98:2d:22:
     94:eb:19:ba:c5:01:11:a5:ae:a9:60:a0:f2:db:4c:aa:9d:f8:
     ee:df:e7:d0:75:8f:da:65:18:6c:f3:7b:67:52:9c:23:99:30:
     74:77:c1:cb:d9:71:45:a5:30:6e:f2:52:e8:c7:9e:44:7d:2c:
     4c:cc:cb:1b:9d:5a:9a:f0:66:72:0f:a6:53:f9:95:1e:09:34:
     d3:16:7e:ac:99:7e:27:eb:5a:2e:f0:29:0a:d0:6d:20:57:b8:
     b7:9f:36:51:10:20:c9:59:68:74:94:e2:13:a4:d1:e9:7f:3c:
     68:b2:b8:55:ca:73:52:23:3d:86:2a:54:31:84:9c:71:47:32:
     f7:af:c4:ab:65:d2:e7:0d:6a:b2:51:f6:f1:1a:af:56:e5:53:
     bd:46:2a:90:fe:2e:23:7e:b5:41:58:f6:a8:71:62:53:d1:ed:
     a3:ea:fb:58
```

Step4: 輸出 Server 憑證本身並做 sha256 的 hash

```
gary@ubuntu:~/Desktop/hw2$ openssl asn1parse -i -in c0.pem -strparse 4 -out c0_body.bin -noout
gary@ubuntu:~/Desktop/hw2$ sha256sum c0_body.bin
1206a04ca64ca62f661db6a2a6d22f624cacb98a75f08200ec551382a7fcdb19  c0_body.bin
gary@ubuntu:~/Desktop/hw2$
```

Step5: 驗章

整理一下目前所的到的資訊:

```
Informations:

Modulus = 9CD30CF05AE52E47B7725D3783B3686330EAD735261925E1BDBE35F170922FB7B84B4105ABA99E350858ECB12AC468870BA3
Exponent e = 10001

Server's certificate signature =
14747ff6db400d76039e169b16dac8f9d7a9
e5bd7c7af1255faf6cff7cf73b4f0231b02d
df569e3b891ec5858e9f3aa6f36504f45026
59862a6f9b977607b2f4dffcaebab1d3f803
6d946c966215f57fa47d9290a19240982d22
94eb19bac50111a5aea960a0f2db4caa9df8
eedfe7d0758fda65186cf37b67529c239930
7477c1cbd97145a5306ef252e8c79e447d2c
4ccccb1b9d5a9af066720fa653f9951e0934
d3167eac997e27eb5a2ef0290ad06d2057b8
b79f36511020c959687494e213a4d1e97f3c
68b2b855ca7352233d862a5431849c714732
f7afc4ab65d2e70d6ab251f6f11aaf56e553
bd462a90fe2e237eb54158f6a8716253d1ed
a3eafb58


Server's certificate hash = 1206a04ca64ca62f661db6a2a6d22f624cacb98a75f08200ec551382a7fcdb19
```

預計: 如果 Server's certificate signature 被(Modulus, Exponent e)這組 Public Key 做解密演算法(也算簽章)之後結果與 Server's certificate hash 的內容相同的話，則代表驗章成功。

程式碼:

```
int main()
{

    BN_CTX *ctx = BN_CTX_new();
    BIGNUM *pk_n = BN_new();
    BIGNUM *pk_e = BN_new();

    BIGNUM *server_signature = BN_new();
    BIGNUM *sign_S2 = BN_new();
    BIGNUM *verify = BN_new();
    BIGNUM *server_certificate_hash = BN_new();

    BN_hex2bn(&server_certificate_hash, "1206a04ca64ca62f661db6a2a6d22f624cacb98a75f08200ec551382a7fcdb19");
    BN_hex2bn(&pk_n, "009CD30CF05AE52E47B7725D3783B3686330EAD735261925E1BDBE35F170922FB7B84B4105ABA99E350858EC
    BN_hex2bn(&pk_e, "010001");
    BN_hex2bn(&server_signature,"14747ff6db400d76039e169b16dac8f9d7a9"
                                "e5bd7c7af1255faf6cff7cf73b4f0231b02d"
                                "df569e3b891ec5858e9f3aa6f36504f45026"
                                "59862a6f9b977607b2f4dffcaebab1d3f803"
                                "6d946c966215f57fa47d9290a19240982d22"
                                "94eb19bac50111a5aea960a0f2db4caa9df8"
                                "eedfe7d0758fda65186cf37b67529c239930"
                                "7477c1cbd97145a5306ef252e8c79e447d2c"
                                "4ccccb1b9d5a9af066720fa653f9951e0934"
                                "d3167eac997e27eb5a2ef0290ad06d2057b8"
                                "b79f36511020c959687494e213a4d1e97f3c"
                                "68b2b855ca7352233d862a5431849c714732"
                                "f7afc4ab65d2e70d6ab251f6f11aaf56e553"
                                "bd462a90fe2e237eb54158f6a8716253d1ed"
                                "a3eafb58");

    BN_mod_exp(verify, server_signature, pk_e, pk_n, ctx);
    printBN("The verify is ", verify);

    return 0;
}
```

輸出結果:

```
gary@ubuntu:~/Desktop/hw2$ ./a.out
The verify is 01FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF003031300D06096086480165030402010500042012
06A04CA64CA62F661DB6A2A6D22F624CACB98A75F08200EC551382A7FCDB19
```

結果發現似乎不完全一樣，一開始還以為自己做錯了，但後來知道是前面會夾帶其他的憑證資訊，後半段經過比對之後與先前得到的 Server’s certificate hash 完全一致，即驗章成功。