

rsa_second

12141163 이육진

Homework

4) Write a program that allows users to generate an RSA key and encrypt/decrypt with this key. Use the fact

$$x*y \bmod n = (x \bmod n)*(y \bmod n) \bmod n$$

For example, we want to compute $a^b \bmod m$. In pseudocode,

```
a1 = a mod m
p=1
for(int i=1;i<=b;i++){
    p *= a1
    p = p mod m
}
```

Now p is the result for $a^b \bmod m$.

To compute $\gcd(a, b)$, use following pseudocode:

```
for(;;){
    if (b==0) break;
    t=b;
    b=a % b;
    a=t;
}
```

Now the resulting a is the $\gcd(a, b)$.

```
93 int compute_phi2(int phi1) {
94     // return num of relative prime numbers to phi1
95     int cnt = 0;
96     for(int i=1;i<phi1;i++)
97         if (GCD(i, phi1) == 1) cnt++;
98     return (cnt);
99 }
100 int GCD(int a, int b) {
101     // return GCD of a, b
102     int t;
103     for (;;)
104     {
105         if (b == 0) break;
106         t = b;
107         b = a % b;
108         a = t;
109     }
110     return (a);
111 }
```

```

~/De/정 /lect8-public-key ./a.out
enter p and q, two prime numbers
11 17
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157
select one of these: 7
(7 23) are ok to use p:11 q:17 n:187 phi1:160 e:7 phi2:64 d:23
enter num to encrypt
144
M:144 C:100
Mp:144

```

수도코드를 참고하여 강의노트의 예제가 정상적으로 계산되는것을 확인할 수 있었습니다.

```

80 int select_e(int phi1) {
81 // display relative prime numbers to phi1
82 // let user select one of them
83     // display relative prime numbers to phi1
84     for(int i=1;i<phi1;i++)
85         if (prime[i])
86             printf("%d ",i);
87     // let user select one of them
88     printf("\nselect one of these: ");
89     int e;
90     scanf("%d", &e);
91     return e;
92 }

```

```

10 char prime[500000];
11
12 int main()
13 {
14     for (int i=2;i<500001;i++)
15         prime[i] = 1;
16     for (int n = 2; n <= floor(sqrt(500000)); n++)
17     {
18         if (!prime[n]) continue;
19         for (int mult = 2; n * mult <= 500000; mult++)
20             prime[n * mult] = 0;
21     }
22 }

```

그중에서도 소수를 활용하는 방법으로 에라토스테네스의 체 방법을 활용하여 50만까지의 소수를 구하고 시작 하였습니다.

```
26      // step 1. compute n
27      int n = p * q;
28      // step 2. compute phi1
29      int phi1 = (p - 1) * (q - 1);
30
31      int e; int phi2; int d;
32
33      for (;;) {
34          // step 3. select e
35          e = select_e(phi1);
36          // step 4. compute phi2
37          phi2 = compute_phi2(phi1);
38
39          // step 5. compute d
40          d = compute_pow(e, phi2 - 1, phi1);
```

7) Install openssl in your pc.

download openssl from <http://gnuwin32.sourceforge.net/packages/openssl.htm>

(get "complete package except source")

install in your pc

go to the installed directory/bin

double click on openssl (you may need to run as "administrator")

```
man3/X509v3_get_ext_by_NID.html
/usr/local/share/doc/openssl/html/man5/config.html
/usr/local/share/doc/openssl/html/man5/x509v3_config.html
/usr/local/share/doc/openssl/html/man7/bio.html
/usr/local/share/doc/openssl/html/man7/crypto.html
/usr/local/share/doc/openssl/html/man7/ct.html
/usr/local/share/doc/openssl/html/man7/des_modes.html
/usr/local/share/doc/openssl/html/man7/Ed25519.html
/usr/local/share/doc/openssl/html/man7/Ed448.html -> /usr/local/share/doc/openssl/html/man7/Ed25519.html
/usr/local/share/doc/openssl/html/man7/evp.html
/usr/local/share/doc/openssl/html/man7/openssl_store-file.html
/usr/local/share/doc/openssl/html/man7/openssl_store.html
/usr/local/share/doc/openssl/html/man7/passphrase-encoding.html
/usr/local/share/doc/openssl/html/man7/proxy-certificates.html
/usr/local/share/doc/openssl/html/man7/RAND.html
/usr/local/share/doc/openssl/html/man7/RAND_DRBG.html
/usr/local/share/doc/openssl/html/man7/RSA-PSS.html
/usr/local/share/doc/openssl/html/man7/scrypt.html
/usr/local/share/doc/openssl/html/man7/SM2.html
/usr/local/share/doc/openssl/html/man7/ssl.html
/usr/local/share/doc/openssl/html/man7/X25519.html
/usr/local/share/doc/openssl/html/man7/X448.html -> /usr/local/share/doc/openssl/html/man7/X25519.html
/usr/local/share/doc/openssl/html/man7/x509.html
```

open ssl 공식 홈페이지를 통하여 다운로드를 받을 수 있었습니다.

8-1) Generate an RSA key pair using openssl.

```
openssl> genrsa -out mykey.pem 1024
```

// generate public/private key pair in file "mykey.pem" with keysize 1024 bits.

// default size 512 bits if keysize is not specified

```
~/De/정 /lect8-public-key openssl
OpenSSL> genrsa -out mykey.pem 1024
Generating RSA private key, 1024 bit long modulus
.....++++++
.++++++
e is 65537 (0x10001)
OpenSSL>
```

open ssl을 통하여 키를 생성하였습니다.

8-2) Convert mykey.pem to a text file, mykey.txt, to look at the contents. Use WordPad to open mykey.txt. Find n, e, and d.

```
openssl> rsa -in mykey.pem -text -out mykey.txt
```

// display the contents of "mykey.pem" in plain text in

// output file "mykey.txt"

```
~/De/정 /lect8-public-key openssl
OpenSSL> rsa -in mykey.pem -text -out mykey.txt
writing RSA key
OpenSSL>
```

mykey.pem 을 mykey.txt로 우리가 볼수 있도록 변환하였습니다.

n: modulus

```
modulus:
00:b5:bb:42:1d:48:77:1d:c6:75:0d:e9:b7:40:15:
ed:57:98:71:34:1e:6c:16:25:45:84:07:e0:b5:1b:
71:f2:04:94:8a:e5:c6:77:69:b5:1e:57:fc:2e:98:
e2:f7:34:76:32:07:3c:9b:dd:70:44:90:4e:ed:de:
17:d4:ee:1d:ca:e9:52:eb:3e:b1:19:f0:20:83:9c:
39:93:dd:cb:20:6e:5e:f1:0f:26:10:96:2a:7f:8b:
96:05:84:50:71:22:60:48:87:26:d4:7c:50:d4:0f:
4c:c9:c2:92:60:2e:dd:cc:23:92:ca:92:cb:6a:ec:
26:34:34:4d:aa:1b:05:d5:6f:
```

e: public exponent

```
publicExponent: 65537 (0x10001)
```

d: private exponent

```
privateExponent:
00:b4:a0:04:5c:7a:93:fa:02:30:81:bd:94:27:9c:
23:a4:76:9d:bd:81:a7:48:73:8f:1e:65:7e:10:43:
d1:03:0d:4e:5e:a7:76:95:65:79:61:49:6e:1b:1a:
56:2c:01:f6:a0:4e:d5:0c:ce:11:31:f4:84:9e:a3:
a9:e8:37:2b:5f:bd:37:36:ee:e5:18:e9:a1:92:bf:
f6:58:b3:ca:12:9f:1e:e6:76:2c:93:8c:25:b9:f5:
71:40:4a:cb:56:d7:ce:98:95:dd:a5:ca:0b:07:fd:
b3:4a:90:b9:a5:3f:d8:75:e5:d5:7f:71:84:26:26:
9b:39:42:f9:ae:2f:05:d4:91:
```

8-3) Encrypt "hello" with (n, e) to produce ciphertext. What is the size of the ciphertext? Decrypt the ciphertext with (n, d) to recover "hello".

```
openssl> rsautl -encrypt -inkey mykey.pem -in myplain.txt -out mycipher // encrypting
openssl> rsautl -decrypt -inkey mykey.pem -in mycipher -out mycipher.dec // decrypting
```

Refer the manual in man/pdf/openssl-mal.pdf.

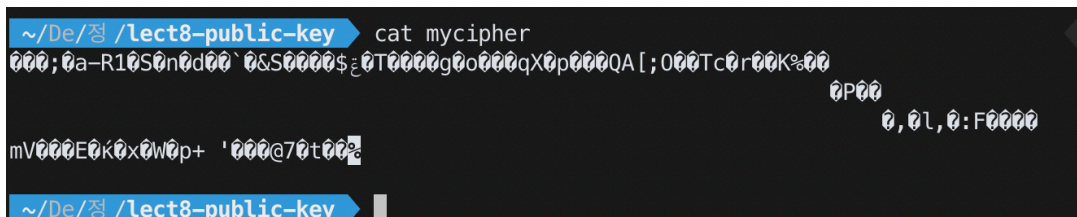
PEM(Privacy Enhanced Mail) file format transforms a binary file into an ascii file using base64. Each 6 bit in the input file will be converted to a letter in {A-Z, a-z, 0-9, +, -}, and wrapped with boundary lines.

ex) Man ==> 77 97 110 ==> 01001101 01100001 01101110
==> T(010011, 19) W(010110, 22) F(000101, 5) u(101110, 46)

(.der : binary DER encoded certificates

.cer : similar to .der

.key : PKCS#8 keys. the keys are encoded as binary DER or ASCII PEM)



```
~/De/정 /lect8-public-key> cat mycipher
000;0a-R10S0n0d00`0&S0000$ε0T0000g0o000qX0p000QA[;000Tc0r00K%00
0P00
0,0L,0:F0000
mV000E0k0x0W0p+ '000@70t00%
~/De/정 /lect8-public-key>
```

처음 encrypt명령어를 통하여 hello 문자를 적어놓은 myplain.txt 를 mycipher로 변환시켰다.
그 후 파일을 열어보니 위와같이 읽은수없는 문자들이 확인되었다.

이어서 decrypt명령어를 실행시켰고 아래와같이 파일이 생성되어 원본의 myplain.txt를 확인할 수 있었다.



```
1 mycipher.dec
1 hello
~
```

9) Make an X.509 certificate.

9.1) make a config file "myconf.txt"

```
[req]
string_mask = nombstr
distinguished_name = req_distinguished_name
prompt = no
[req_distinguished_name]
commonName = my CA
stateOrProvinceName = some state
countryName = US
emailAddress = root@somename.somewhere.com
organizationName = mycompany
```

9.2) Make a certificate for the person/company specified in myconf.txt. The public key of this person/company is given in mykey.pem:

```
req -config myconf.txt -new -x509 -key mykey.pem -out mycert.pem
```

9.3) let's read the contents of the certificate

```
x509 -in mycert.pem -text -out mycert.txt
```

Who is the owner of this certificate? What is the public key? What is the key size?

Who has signed this certificate?

10) Get a certificate in Internet Explorer or in Chrome. Check the contents of x.509 file.

10.1) Go to "tools>internet options>contents>certificates" to get a copy of a certificate. To view the certificate of a site:

In Chrome, go to some https site such as www.daum.net and select Three Dots Menu>More Tools>Developer Tools>Security>View certificate>Details>Copy to File

In IE, go to some https site such as www.daum.net and click the padlock symbol, select "View Certificate">Detail>Copy to File.

10.2) Look at the contents of this certificate (assume the file name is daum.cer) with

```
x509 -in daum.cer -text -out daum.cer.txt -inform DER
```

Who is the owner of this certificate? Who has signed this certificate? What is the public key? What is the key size?

(DER: Distinguished Encoding Rules)