rsa_final

12141163 이욱진

Homework

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

x*y mod n = (x mod n)*(y mod n) mod n

For example, we want to compute a^b mod 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 mod 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).

```
int compute_phi2(int phi1) {
    // return num of relative prime numbers to phi1
    int cnt = 0;
    for(int i=1;i<phi1;i++)
        if (GCD(i, phi1) == 1) cnt++;
    return (cnt);
}
int GCD(int a, int b) {
    // return GCD of a, b
    int t;
    for (;;)
    {
        if (b == 0) break;
        t = b;
        b = a % b;
        a = t;
    }
    return (a);
}
```

```
~/De/정 /lect8-public-key  ./a.out                                      ✔  11:43:23
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 10
7 109 113 127 131 137 139 149 151 157
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.

```c
int select_e(int phi1) {
// display relative prime numbers to phi1
// let user select one of them
            // display relative prime numbers to phi1
    for(int i=1;i<phi1;i++)
        if (prime[i])
            printf("%d ",i);
            // let user select one of them
    printf("\nselect one of these: ");
    int e;
    scanf("%d", &e);
    return e;
}
```

```c
char prime[500000];

int main()
{
    for (int i=2;i<500001;i++)
        prime[i] = 1;
    for (int n = 2; n <= floor(sqrt(500000)); n++)
    {
        if (!prime[n]) continue;
        for (int mult = 2; n * mult <= 500000; mult++)
            prime[n * mult] = 0;
    }
```

그중에서도 소수를 활용하는방법으로 에라토스테네스의체 방법을 활용하여 50만까지의 소수를 구하고 시작하였습니다.

```
        // step 1. compute n
    int n = p * q;
       // step 2. compute phi1
    int phi1 = (p - 1) * (q - 1);

    int e; int phi2; int d;

    for (;;) {
                  // step 3. select e
        e = select_e(phi1);
                  // step 4. compute phi2
        phi2 = compute_phi2(phi1);

                  // step 5. compute d
        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.ht
ml
/usr/local/share/doc/openssl/html/man7/evp.html
/usr/local/share/doc/openssl/html/man7/ossl_store-file.html
/usr/local/share/doc/openssl/html/man7/ossl_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".

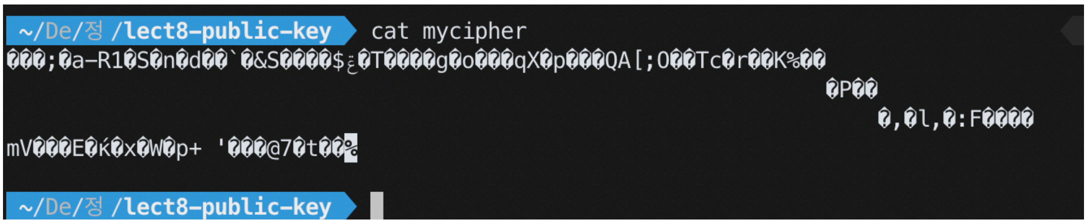
openssl> rsautl -encrypt -inkey mykey.pem -in myplain.txt -out mycipher // encrypting
openssl> rsautl -decrypt -inkey mykey.pem -in mycipher -out mycipher.dec // decrypting

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)



처음 encrypt명령어를 통하여 hello 문자를 적어놓은 myplain.txt 를 mycycipher로 변환시켰다.
그 후 파일을열어보니 위와같이 읽은수없는 문자들이 확인되었다.

이어서 decrypt명령어를 실행시켰고 아래와같이 파일이생성되어 원본의 myplain.txt를 확인할 수 있었다.

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

```
1 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
```

myconf.txt 를 만들어주었고 내용을 그대로 넣어주었습니다.

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

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

주어진 명령어를 실행하여 인증서 mycert.pem을 만들었습니다.

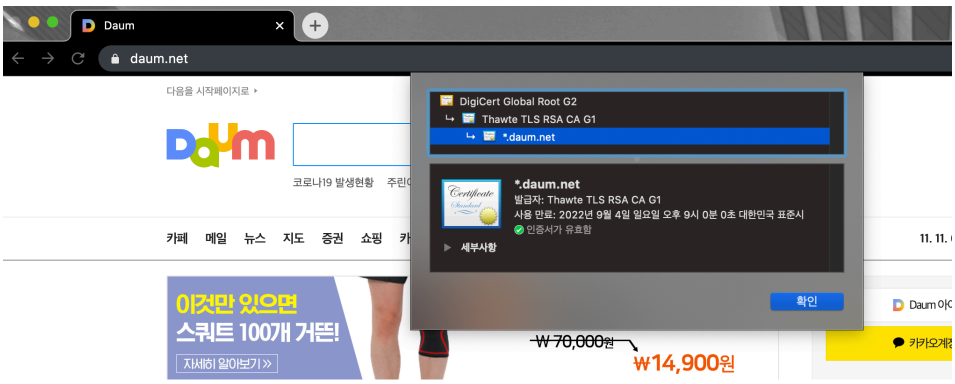
9.3) let's read the contents of the certificate
x509 -in mycert.pem -text -out mycert.txt
Who is the owner of this certificate? What is the public key? What is the key size?
Who has signed this certificate?

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

```
mycert.txt
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            23:50:79:80:c4:c7:a9:29:ef:e7:07:ac:4a:ae:40:9f:af:2f:00:c2
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = my CA, ST = some state, C = US, emailAddress = root@somename.somewhere.com, O = mycompany
        Validity
            Not Before: Nov 11 09:45:24 2020 GMT
            Not After : Dec 11 09:45:24 2020 GMT
        Subject: CN = my CA, ST = some state, C = US, emailAddress = root@somename.somewhere.com, O = mycompany
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (1024 bit)
                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
```

이 인증서의 소유자는 subject의 O부분인 mycompany이고 공개키는 Modulus 부분이다. 키의 사이즈는 이전의 우리가 설정했던 1024 비트이고 인증서의 서명은 역시나 Issuer의 O부분인 mycompany이다.

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.



daum에 접속하여 *.daum.net.cer 을 복사하였습니다.
그 후 이름을 daum.cer 로 수정하였습니다.

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)

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

위 명령어를 실행하였고 txt파일을 읽어보았습니다.

```
1 daum.cer.txt
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            02:62:36:d6:50:aa:3b:f5:d4:73:8b:98:a8:93:f4:c4
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = DigiCert Inc, OU = www.digicert.com, CN = Thawte TLS RSA CA G1
        Validity
            Not Before: Jun  5 00:00:00 2020 GMT
            Not After : Sep  4 12:00:00 2022 GMT
        Subject: C = KR, ST = Jeju-do, L = Jeju-si, O = Kakao Corp., CN = *.daum.net
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:e9:e3:dc:04:68:c6:33:2b:59:db:37:b7:35:01:
                    2a:71:14:12:99:27:ad:f1:6e:61:e3:54:84:42:c5:
                    27:02:bd:3f:28:28:d5:72:03:d5:a9:68:f3:31:45:
                    99:e3:54:67:45:1c:9b:fa:ce:9c:0e:71:e8:e1:47:
                    e9:c9:ea:35:48:c2:92:50:68:47:24:ef:2a:f4:1f:
                    98:6b:11:3b:50:70:d9:38:28:e3:07:9e:f7:de:de:
                    eb:1e:85:33:9d:e6:f7:bb:0e:51:3f:9a:95:6f:95:
                    b5:3d:23:a4:15:5f:c7:ad:db:4d:fc:72:fd:80:36:
                    f2:e8:8b:a7:b8:42:05:b3:6e:bc:2c:ec:20:91:51:
                    93:33:d4:eb:d3:1d:09:30:89:07:11:b9:bd:c0:11:
                    80:bf:00:ec:93:e1:82:67:d7:8c:9e:a0:5a:4b:a9:
                    b5:97:39:08:54:25:e3:c0:c1:66:39:a0:36:89:cc:
                    cb:7e:37:88:b7:42:21:a3:59:58:3c:76:05:d3:9b:
                    46:81:36:40:1c:97:b0:e4:5d:fc:b4:22:91:57:04:
                    3c:6b:84:55:39:1b:e4:7c:48:5a:01:8a:a8:de:fa:
                    8a:77:f5:a6:5a:27:da:fb:02:dc:f3:d9:ed:f0:17:
                    33:b4:96:bc:c3:c4:a6:57:34:d2:2b:8d:eb:96:c9:
                    ac:1b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:A5:8C:FE:32:CC:EB:0F:2C:D4:19:C6:08:B8:00:24:88:5D:C3:C5:B7
```

위 사진으로 알 수 있듯이 인증서의 owner는 카카오 Corp이며 이 인증서에 사인한 사람은 Thawte TLS RSA CA G1 이다.
공개키로는 2048비트의 크기를 가지고있으며 위사진의 Modulus: 부분의 해당한다.