rsa_first

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.

```
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;
}
```

```
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")

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
```

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"
```

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)

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)