[수업내용]

수업내용을 한번더 복습 후 왼전히 이해한 다음, 과제를 시작하도록 하겠습니다.

Q. head.h라는 파일에

#include <stdio.h>

extern int ex\_value;

util.c라는 파일은

#include "head.h"

int ex\_value=10;

main.c 라는 파일에선

#include "head.h"

void main(){

printf("ex\_value:%d",ex\_value);

}

라고 코드를 작성하였는데 비쥬얼 스튜디오에선 정상적으로 hello가 출력이 되는데 putty에선 gcc -o main main.c 로 컴파일 하여도 undefinde reference to 'ex\_value'라고 하며 컴파일이 되지 않습니다. 혹시 gcc에 옵션을 추가를 해야하는게 있나요?

A. 모든 c 파일을 동시에 컴파일해야 합니다. gcc -o main main.c util.c

util.cpp has helper functions. For example, you can use rotate\_left\_and\_store() function in util.cpp for comp\_Ci\_Di().

[HOMEWORK]

1) Implement Caesar's cipher system. Write a program that breaks this system.

텍스트이(가) 표시된 사진

자동 생성된 설명

Key\_Schedule 코드는 위와 같습니다.

각각 PC\_1을 통해서 K를 K+로,   
ROL을 통해서 K+를 각각의 C 0~16 , D 0~16으로 만듭니다.   
마지막으로 PC\_2를 이용해 keys 16개를 만들어서 저장합니다.

테이블이(가) 표시된 사진

자동 생성된 설명

2) Implement DES

ref: "The DES Algorithm Illustrated" :

<http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm>

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Encrypt 코드는 위와 같습니다.

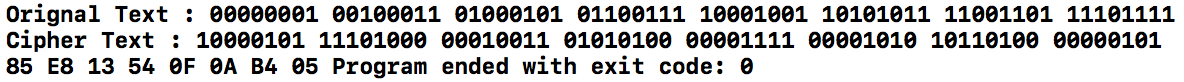
이 코드에 대해 설명하자면,   
차례대로 M 이라는 Plain Text가 있으면, 이것을 IP를 통해서 한번 비트를 섞은 M+를 만들고, 이를 L, R배열로 각각 분리 시킵니다..  
 Ln는 Rn-1이 되고, 남은 Rn배열은 Rn-1배열을 E bit Selection을 이용해서 48bit 배열로 바꾸고, 이 48bit로 바꾼 배열 key\_schedule을 통해 만들어 낸 k와 XOR연산을 시행합니다 그 후 배열을 6bit B배열 8갤 바꾸어 저장한 뒤,   
Sbox에 각각 넣어 SB 배열을 만든후, 배열 SB를 P박스에 넣어서 32bit짜리 배열로 다시 만들어줍니다..  
 그 뒤 Ln-1과 XOR 연산을 실행, Rn 배열이 됩ㄴ;다.   
이렇게 16개의 L, R 배열을 만들어, L16, R16을 IP\_1 연산을 통해 마지막으로 섞어준 것은 Cipher Text가 됩니다.

테이블이(가) 표시된 사진

자동 생성된 설명

녹색, 건물, 그룹, 컴퓨터이(가) 표시된 사진

자동 생성된 설명



3) Make an RSA key pair from two small prime numbers (p=11, q=3). Encrypt/decrypt for some example numbers. The example number should be in the range of 0 to 32. Try to encrypt HELLO using this RSA key.

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).

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

\t

텍스트이(가) 표시된 사진

자동 생성된 설명

RSA는 비대칭 키 알고리즘을 사용합니다.

두 소수 p, q를 선택해, n=p\*q, phi1 = (p-1)\*(q-1), 를 구현합니다.  
만들어진 phi에서 phi와 서로소인 phi보다 작은 수 중에 하나를 선택합니다. =e.   
phi2는 phi1 보다 작은 수 중 phi1과 서로소인 수의 개수를 나타내고, 이렇게 구해진 phi2를 이용해 d=ephi2-**1mod** phi1 을 하여 구할 수 있습니다.   
: Public Key = (n, e) , Private Key = (n, d) 입니다.

5) Write a program that breaks your RSA system.

6) Write a pair of secure client and server that exchange a password using RSA system.

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")

텍스트이(가) 표시된 사진

자동 생성된 설명

open ssl 을 설치해주었습니다.

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](http://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](http://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)